MARCEL HENROT . JACQUES BOISGONTIER



EDICIONES ELISA, S. A.

ZX-SPECTRUM
para todos

OTRAS OBRAS DEL FONDO EDITORIAL
DE
EDICIONES ELISA, S.A.

DICCIONARIO DEL BASIC, por **D. A. Lien.**

EL DESCUBRIMIENTO DEL COMMODORE, 64, por **D. J. David.**

102 PROGRAMAS PARA ZX81 Y SPECTRUM, por **J. Deconchat.**

102 PROGRAMAS PARA COMMODORE 84, por **J. Deconchat.**

CLAVES PARA EL APPLE II, APPLE II PLUS y APPLE IIe, por **N. Breaud-Pouliquen.**

PASAPORTE PARA APPLESOFT, por **C. Galais.**

EL APPLE Y SUS FICHEROS, por **J. Boisgontier**

COMMODORE 64 PARA TODOS, por **J. Boisgontier, S. Brébion** y **G. Foucault**

ZX-SPECTRUM para todos

por

Marcel Henrot

y

Jacques Boisgontier

Traducción:

EQUIPO DE INFORMÁTICA EXPERIMENTAL
APLICADA (EINEA)



EDICIONES ELISA, S.A.

1985

Título original de la obra: SPECTRUM POUR TOUS

Primera edición: julio 1985.

ISBN: 84-7622-008-1.
Depósito legal: B. 22.670-1985.

Printed in Spain Impreso en España

GRAFFING, S.A. - Arquímedes, 18 - HOSPITALET DE LLOBREGAT

SUMARIO

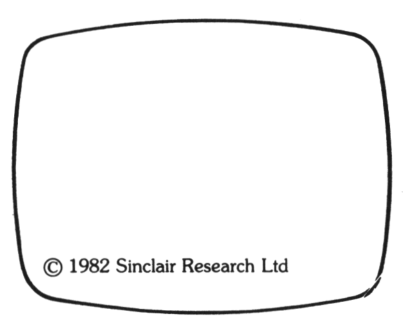
Toma de contacto con su Spectrum	**7**
Estructura de la pantalla y visualización	**14**
Programa y edición	**20**
Atributos	**23**
Introducción de variables por programación	**27**
Variables del sistema	**29**
Definición de funciones	**32**
Circuitos (bucles)	**36**
Cadenas de caracteres	**40**
Gráficos definidos por el usuario	**45**
Introducción de parámetros por programa	**47**
Captación de caracteres	**53**
Alta definición	**56**
Sonidos	**60**
Algunos dibujos	**63**
Rotaciones y desplazamientos	**68**
Iluminación	**73**
Minilogo	**77**
ANEXO 1: Recapitulación de las instrucciones BASIC	**85**
ANEXO 2: Recapitulación de las funciones	**99**
ANEXO 3: Códigos de los caracteres	**111**
ANEXO 4: Informe de errores	**115**
ANEXO 5: Las variables del sistema	**120**

TOMA DE CONTACTO CON SU SPECTRUM

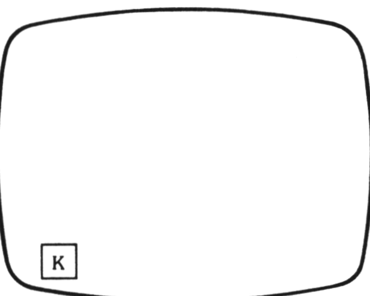
Para dialogar con su ordenador usted dispone de un teclado; con él puede darle órdenes e instrucciones y él le responderá utilizando la pantalla de un televisor.
Este diálogo es apasionante.
Si usted habla en castellano a un esquimal, éste no le comprenderá. Del mismo modo, el ordenador sólo comprenderá el lenguaje que tiene establecido: **el Basic.** Confíe en él, le señalará los errores de sintaxis que usted pueda cometer al introducir sus instrucciones.
El Spectrum posee numerosas instrucciones y funciones. Las principales se desarrollan dentro de esta obra y todas se repiten en un anexo, acompañadas de comentarios y ejemplos.
En el anexo encontrará también importantes notas para la utilización de las variables del sistema, tales como la redefinición de uno o varios nuevos juegos de caracteres, la extensión de los gráficos definidos por el usuario en más de 21 caracteres, etc.

El teclado

En el momento en que se conecta el ordenador, aparece un mensaje en la parte inferior de la pantalla.



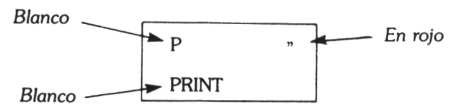
Pulse la tecla **SPACE** y verá aparecer, en el ángulo inferior izquierdo de la pantalla, el cursor K papadeante.



El cursor K

El cursor K le indica que debe introducir una palabra clave (Key).
Hay 50 palabras clave a su disposición; cada una representa una instrucción.

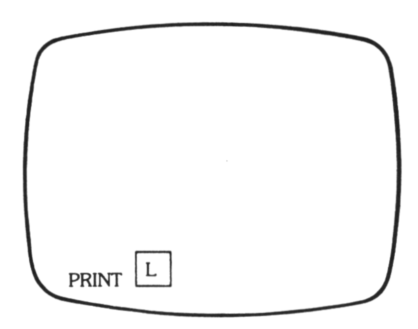
En las 26 teclas alfabéticas de su teclado, encontrará 3 inscripciones; así, por ejemplo, en la tecla P



encontrará la letra P en blanco, el signo comillas " en rojo y la palabra PRINT en blanco.

En modalidad K se visualiza la palabra.

Pulse P y verá

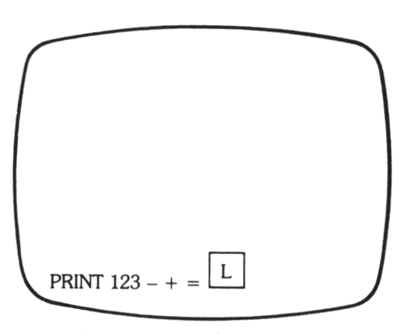


El cursor L

El cursor L, que aparece después de la palabra PRINT, indica que se encuentra usted en modalidad L y que debe introducir una letra (Letter), un signo o una cifra; digamos simplemente un carácter.

Si añadimos las 10 teclas numéricas, disponemos de 36 teclas. Usted puede visualizar cualquiera de estas 36 teclas pulsándola. **Para visualizar el carácter o la palabra impresa en rojo sobre una de estas 36 teclas, pulse la tecla SYMBOL SHIFT (impresa en rojo) y, sin liberarla, pulse una de las 36 teclas.**

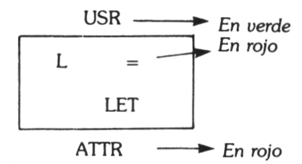
Por ejemplo, pulse 1, después 2, luego 3, después, manteniendo SYMBOL SHIFT, teclee sucesivamente J, K, L y verá:



El cursor [E]

Veamos una de las múltiples funciones de la tecla **CAPS SHIFT.** Si usted pulsa simultáneamente CAPS SHIFT y SYMBOL SHIFT, el cursor [L] se convierte en [E]. **Se encuentra usted en modalidad E (Ampliada): amplía el significado de las teclas a las palabras y signos impresos fuera de ellas.**

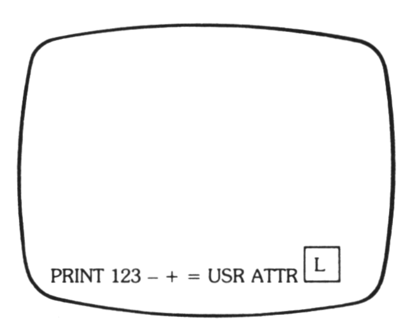
Por ejemplo:



Encima de las 26 teclas alfabéticas, hay 26 palabras impresas en verde. Se visualizará una de ellas. Pulse L: obtendrá la visualización de USR; después, otra vez el cursor [L].

Volvamos a la modalidad E. Debajo de las 36 teclas alfanuméricas, hay 36 palabras o signos. Para visualizar uno de ellos, pulse simultáneamente SYMBOL SHIFT y una de las 36 teclas.

Pulse SYMBOL SHIFT y L: obtendrá ATTR y otra vez el cursor [L].



Para volver a [L] sin utilizar la modalidad [E], pulse una 2.ª vez las dos teclas CAPS SHIFT y SYMBOL SHIFT.

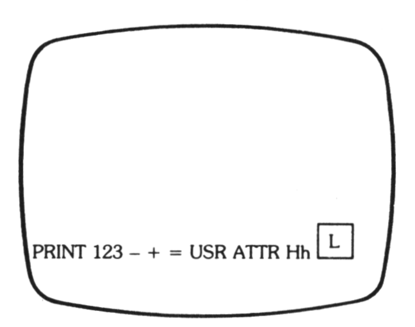
El cursor [C]

Pulsando simultáneamente CAPS SHIFT y una de las 10 teclas numéricas, obtendremos la función escrita en blanco encima de estas teclas.

Pulse CAPS SHIFT y, manteniéndola, pulse **CAPS LOCK** (tecla 2): el cursor [L] se reemplazará por el cursor [C].
Se encuentra usted en modalidad C (Capital), o modalidad de mayúsculas.
Pulse H: obtendrá H y seguirá con el cursor [C]
Pulse una 2.ª vez CAPS SHIFT y 2: volverá a [L].

Pulse H: obtendrá h.

En este momento, la pantalla se presenta así:



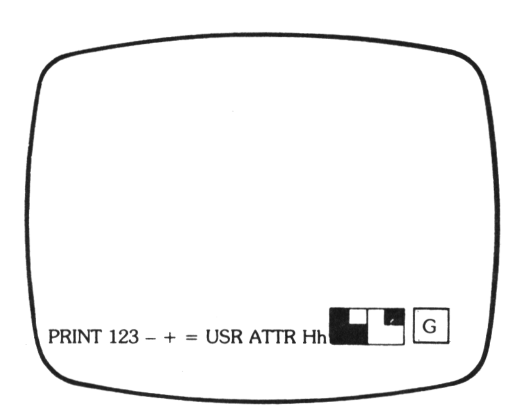
El cursor G

Pulsando CAPS SHIFT y, al mismo tiempo, **GRAPHICS** (tecla 9), el cursor L se reemplazará por el cursor G.
Se encuentra usted en modalidad G (Gráfica).

Sobre las teclas numéricas 1 a 8, además de las cifras y de los signos impresos en rojo, existen 8 caracteres gráficos predefinidos. Se visualizan cuando se pulsa CAPS SHIFT y, al mismo tiempo, una de las teclas 1 a 8.
Pulse CAPS SHIFT y 1: obtendrá ▙.

Si usted no pulsa CAPS SHIFT, pero sí 1, obtendrá el mismo carácter, pero invertido.

Pulse 1: obtendrá ▝.
He aquí la pantalla:

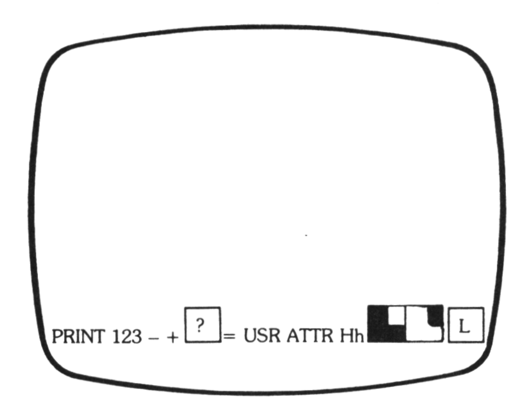


Para regresar a la modalidad L, pulse una 2.ª vez CAPS SHIFT y 9.

El indicador [?]

Hemos introducido diferentes palabras y caracteres, que aparecen en la parte inferior de la pantalla, llamada zona de edición. **Para introducir esta línea en el ordenador y hacer que éste la ejecute, es normal pulsar ENTER.**

Pulse ENTER y mire la pantalla:

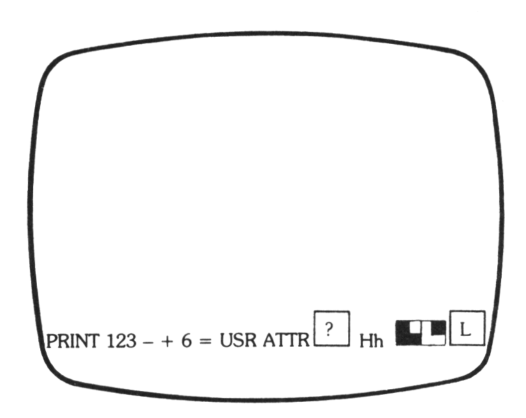


El Spectrum le advierte, mediante el indicador parpadeante [?], que ha cometido usted un error de sintaxis y que rechaza su línea. También indica dónde se sitúa este error: después del carácter +.

Corrijamos este error:
Para desplazar el cursor [L] hacia la izquierda, pulse CAPS SHIFT y ⇦ (tecla 5) para trasladar, poco a poco, el cursor entre el carácter + y el carácter =.
Pulse la tecla 6.

Para llevar el cursor [L] al final de la línea, pulse CAPS SHIFT y ⇨ (tecla 8), dejando el dedo sobre la tecla 8 (todas las teclas son de repetición, si se las mantiene pulsadas).
Pulse ENTER.

Ésta es la pantalla:



El Spectrum rechaza su línea e indica un error de sintaxis delante del carácter H. El ordenador señala, de esta manera, todos los errores, hasta que todos ellos hayan sido corregidos. Entonces, aceptará su línea y la ejecutará.

La función DELETE

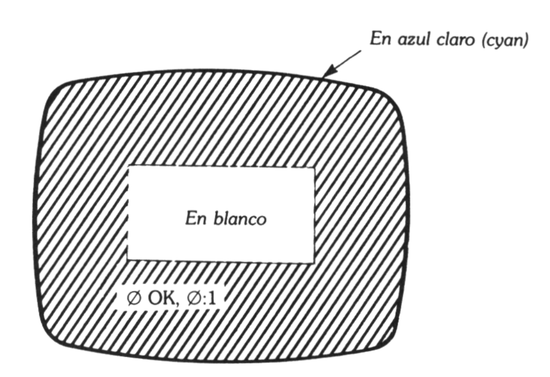
Veamos otra función de CAPS SHIFT (las otras se verán en capítulos posteriores). Pulse CAPS SHIFT y **DELETE** (tecla 0). **Verá como se borran todos los caracteres y palabras, uno después de otro,** si se deja la tecla pulsada, o uno a uno, a cada pulsación.
Al final sólo quedará el cursor K.

Nota: La visualización y el borrado de las palabras (impresas en las teclas o fuera de ellas) son dos de las particularidades originales del Spectrum. Estas palabras se codifican con un solo número, lo que permite que su manejo sea más fácil para la MEM (ROM) del ordenador y para sus propios programas (especialmente en códigos de máquina). Evidentemente, el teclado es complejo; pero con la costumbre, que se adquiere rápidamente, notará usted la comodidad que este sistema aporta. Con este sistema también se evitan errores; usted no podría introducir PRONT por PRINT.
Ejercítese a introducir los caracteres o las palabras que desee. El ordenador no corre ningún peligro.

La pantalla

La pantalla es el medio privilegiado del ordenador para responderle.

Previamente, acabemos el examen del teclado por las palabras impresas en color, encima de las teclas 1 a 7 y 0. Es un recordatorio de los 8 posibles colores de su ordenador. Las cifras 0 a 7 sirven de argumento a las instrucciones que visualizan estos colores.
Pulse las teclas para hacer: BORDER 5 [L]
Pulse ENTER,
He aquí la pantalla

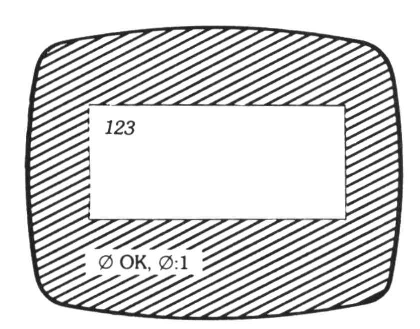


La pantalla utilizada por el ordenador es más pequeña que la de su televisor. Usted ve aquí sus límites físicos.

Observe igualmente que, cuando el ordenador ha aceptado su línea, la ejecuta y, después, visualiza un informe. Éste empieza por 0, que es el código del informe (ver anexo 4), seguido

del texto asociado OK, para señalar: todo va bien. Después 0:1, que significa línea 0 (o mandato, si no, usted obtendría el número de línea del programa) y :1, que significa declaración 1 ejecutada (el ordenador admite varias declaraciones por línea).

Pulse las teclas para efectuar: PRINT 123 L
después, ENTER



La pantalla se divide en dos zonas: **la zona de visualización,** donde está visualizado el número 123, y **la zona de edición,** donde se encuentra el informe. En la inicialización, la zona de visualización contiene 22 líneas y la zona de edición 2 líneas.

Observe así mismo que **la instrucción BORDER asigna el color de los márgenes de la pantalla, así como el color de fondo de la zona de edición.**

Pulse: BORDER 0 ENTER

El color se vuelve negro y el informe se visualiza en blanco. El ordenador emplea, automáticamente, un color que contraste, para visualizar en la zona de edición. El color será blanco si se utiliza un color de fondo oscuro (de 0 a 3) y negro para un color de fondo claro (de 4 a 7).

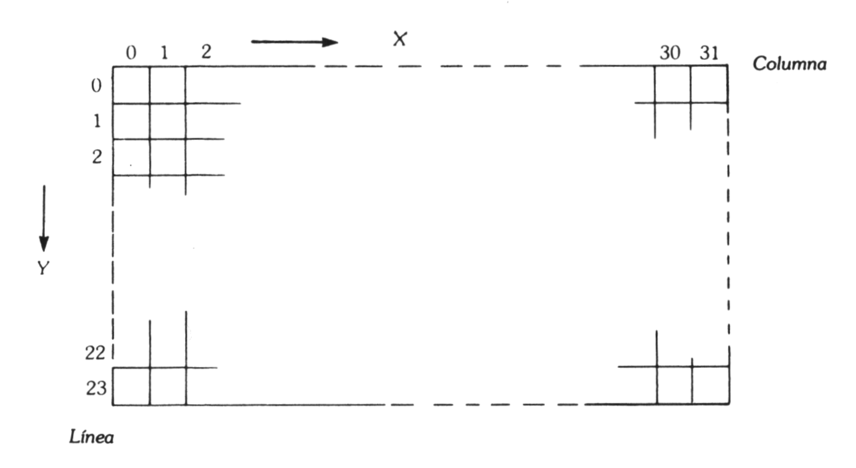
En resumen

- *Para la utilización del teclado, usted dispone de* **5 modalidades K, L, E, C** *y* **G** *y de* **2 teclas especiales CAPS SHIFT** *y* **SYMBOL SHIFT.**
- *Para señalar sus errores de sintaxis, el Spectrum utiliza el indicador* [?], *a menudo llamado marcador por los autores.*
- *Para corregir, usted desplazará el cursor en los dos sentidos horizontales ⇦ ⇨.*
- *Para borrar, usted utilizará* **DELETE.**
- *La pantalla de 24 líneas se divide en dos zonas: la zona de visualización, normalmente de 22 líneas, y la zona de edición, normalmente de 2 líneas. Señalemos que estas dos líneas pueden ampliarse en algunos casos.*
- **BORDER** *asigna el color de los márgenes y de la zona de edición.*

ESTRUCTURA DE LA PANTALLA Y VISUALIZACIÓN

Líneas y columnas

La pantalla se divide en 24 líneas, numeradas de 0 a 23, **y en 32 columnas,** numeradas de 0 a 31, de la manera siguiente:



Existen, pues, 24 × 32, es decir 768 posiciones de visualización, o posiciones PRINT. **La instrucción PRINT sirve para visualizar en la pantalla.** Con esta instrucción sólo puede visualizarse en las 22 líneas superiores.

Para borrar la pantalla, se utiliza la instrucción CLS, que, al mismo tiempo, vuelve a poner la posición PRINT a (0,0).

Pulse las teclas para obtener la secuencia siguiente:

```
CLS             ENTER
PRINT 123       ENTER
PRINT 95437     ENTER
PRINT PI        ENTER
```

Ésta es la pantalla:

```
123
95437
3.1415927
```

Cada vez que utiliza la instrucción PRINT, se desplaza la posición PRINT a la línea siguiente.
Teclee la secuencia:

```
CLS               ENTER
PRINT 123;        ENTER
PRINT 456         ENTER
```

Ésta es la pantalla:

```
123456
```

así pues, el ; (punto y coma) no desplaza la posición PRINT.
Teclee lo siguiente:

```
CLS               ENTER
PRINT 123,456     ENTER
```

y la pantalla visualiza:

```
123               456
```

así pues, la , (coma) desplaza la visualización 16 columnas.
Teclee esto:

```
CLS               ENTER
PRINT 123'456     ENTER
```

ésta es la visualización:

```
123
456
```

así pues, el apóstrofo ' desplaza la posición PRINT a la línea siguiente, lo que hace PRINT sin argumentos.
Introduzca

```
CLS                         ENTER
PRINT 123                   ENTER
PRINT                       ENTER
PRINT                       ENTER
PRINT 456                   ENTER
```

y usted obtendrá

```
123

456
```

Habría obtenido el mismo resultado efectuando:

```
PRINT 123'''456        ENTER
```

Para visualizar una cadena de caracteres, ésta debe estar situada entre comillas.
Teclee:

```
CLS                          ENTER
PRINT "buenos días"          ENTER
```

Los 3 signos ; , y ' son de aplicación

```
PRINT "buenos";"días"
```

facilita el mismo resultado.

PRINT AT y,x;

Para visualizar en un lugar cualquiera de la pantalla, se utilizan las dos letras **AT seguidas de 2 parámetros.** El primero :y es el número de línea (de 0 a 21), y el segundo :x es el número de columna (de 0 a 31). Intente:

```
CLS                          ENTER
PRINT AT 10,10;123;AT 20,20;456       ENTER
```

Si usted efectúa:

```
PRINT AT 0,31;123
```

Tendrá 1 en la columna 31 de la línea 0, y 23 en las columnas 0 y 1 de la línea 1.

He aquí una combinación divertida:
Introduzca (ya sabe que hay que pulsar ENTER al final de cada línea)

```
CLS
PRINT "HASTA LA VISTA"
PRINT AT 21,31;"  "
```

(ponga 2 espacios entre las comillas)

y el texto de la línea 0 desaparece.
¿Qué ha ocurrido?

Al colocar dos espacios en (21,31), uno se ha situado en (21,31) y el otro ha querido situarse en (22,0); pero como no es posible, se ha situado en (21,0), haciendo desfilar (scroll) la pantalla una línea hacia arriba, y la línea 0 ha desaparecido.

Haga además esto:

```
CLS
PRINT AT 21,31;"  "
```

(2 espacios)

el ordenador le responderá:

```
scroll?
```

Usted ve por qué el ordenador provoca un desfile cuando existe una visualización en la pantalla.

PRINT TAB x;

Cuando se desea visualizar en una columna x, se utiliza la palabra TAB seguida del parámetro x.
Efectúe:

```
CLS
PRINT TAB 16;123
```

El Spectrum mantiene el parámetro de TAB en 2 octetos. Por consiguiente, para este parámetro, puede usted poner un número de 0 a 65535. Cualquiera que sea el número que usted introduzca como parámetro, el ordenador lo reduce al módulo 32 y utiliza el residuo como columna de visualización.

Introduzca:

```
CLS
PRINT TAB 32;1
PRINT TAB 54;2
PRINT TAB 256;3
```

Así 0, 32, 64, 256 visualizarán en la columna 0.
Lo mismo 1, 33, 65, 97 visualizarán en la columna 1.
Esto es útil en los cálculos de posición.

Calculadora

Utilizar el ordenador como una calculadora no ofrece ninguna dificultad. Utilice PRINT, para visualizar sus cálculos.

Las funciones +, –, *, /, ↑ dan la adición, la substracción, la multiplicación, la división y la elevación a una potencia.

Efectúe:

```
PRINT 43+45          88
PRINT 126/9          14
PRINT 2*PI           6.2831853
PRINT 2↑8            256
```

La función SQR proporciona la raíz cuadrada:

```
PRINT SQR 64         8
```

pero también:

```
PRINT 64↑.5          8
```

da el mismo resultado.

Las funciones LN y EXP proporcionan los logaritmos neperianos y las exponenciales de base e.

Para obtener el valor de e, efectúe

```
PRINT EXP 1          2.7182818
```

Las funciones circulares disponibles: SIN, COS, TAN, ASN, ACS, ATN, se utilizan con frecuencia.

```
PRINT COS PI         - 1
```

De esta manera, usted dispone de tablas de logaritmos y de tablas trigonométricas de utilización agradable.

Introduzca una cadena:

```
PRINT "2+3=";7
```

Si su cónyuge pasara en este momento, ella (o él) le diría que su ordenador no vale nada. Para tranquilizarla (lo), efectúe:

```
PRINT "6-4=";6-4
```

La utilización de paréntesis cambia las prioridades.

```
PRINT (2+8)/2
PRINT 2+8/2
```

LET v=x

La instrucción LET sirve para asignar un valor x a una variable v. El nombre de una variable debe empezar siempre con una letra. De esta manera, usted dispone de numerosas "memorias", en mucho mayor número de las disponibles en una calculadora de bolsillo.

Para el almacenamiento en la memoria, efectúe:

```
LET a1= 12/100
```

Para volver a llamarla, utilice a1 (sin comillas, ya que se trata de una variable numérica y no de una cadena).

```
PRINT 360*a1
```

que proporciona el 12% de 360.

Si usted necesita otras "memorias", otras variables en lenguaje ordenador, denomínelas a2, a3, b, parís, querida, cocodrilo, etc.

No utilice nunca una variable que no hubiera sido definida con anterioridad. El ordenador la refuta por completo.

En resumen

- *La pantalla se divide en 768 posiciones de visualización o posiciones PRINT.*
- *La instrucción* **CLS** *pone la pantalla a blancos.*
- *La instrucción* **PRINT** *visualiza números o cadenas de caracteres. La posición PRINT se desplaza con la utilización de signos y de palabras; , ' AT y,x; TAB x;*
- *El ordenador puede utilizarse como una calculadora eficaz.*
- *La instrucción* **LET** *asigna un valor a una variable.*

PROGRAMA Y EDICIÓN

Marcador >

Si colocamos un número delante de la instrucción, el mandato representado se convierte en una declaración o línea de programa. De esta manera:

```
10 LET a1=12/100
```

Si la línea es correcta, bajo el punto de vista de la sintaxis, aparecerá abajo de la pantalla, pulsando ENTER.

```
10>LET a1=12/100
```

El indicador, o marcador > , indica la línea que usted acaba de introducir, la cual acaba de editarse.
Introduzca:

```
20 LET a2=360
30 PRINT a1*a2
```

Su programa se compone de 3 líneas. Pulse:

```
RUN
```

y la pantalla visualizará 43.2
Pulsando ENTER, usted lista automáticamente su programa.
La pantalla visualiza:

```
10 LET a1=12/100
20 LET a2=360
30>PRINT a1*a2
```

Para cambiar la línea 10, debe usted desplazar el indicador > a esta línea, pulsando CAPS SHIFT y ⇧ (tecla 7), 2 veces seguidas. Cuando el indicador esté en la línea 10, pulse CAPS SHIFT y EDIT (tecla 1), y la línea 10 aparecerá en la zona de edición.

Usted desea calcular una TVA de 25%, desplace el cursor L a la derecha de la cifra 2 (con CAPS SHIFT y la tecla 8), pulse 2 veces DELETE: el número 12 desaparecerá. Pulse 2,

después 5, por último ENTER: su línea 10 ha cambiado.
RUN
y el ordenador visualiza 90.

Los recursos de edición y de corrección de una línea de programa son flexibles y eficaces. **En cuanto usted introduce una línea de programa que lleva un número ya existente, la línea antigua se borra y se reemplaza por la nueva. Una nueva línea se intercala entre dos líneas de número más pequeño y más grande.**

El intérprete que reside en su Spectrum "interpreta" la línea editada, en el momento en que usted la introduce. Al principio, el intérprete verifica la sintaxis; rechaza la línea si existe un error, señalando el lugar donde se sitúa.

Este controlador de sintaxis muy potente evita muchas decepciones a los programadores principiantes (así como a los más veteranos, ya que la utilización de una instrucción o de una función, sin verificar la sintaxis en el manual, es a menudo el origen de un error así señalado).

A continuación, el intérprete ejecuta la línea editada, si se trata de un mandato, o la guarda en la zona de programa, en el lugar adecuado, si se trata de una línea de programa.

Pulsando RUN, el intérprete ejecuta todas las líneas del programa, una después de otra, en orden numérico creciente.

Si existe un error de programación, la ejecución se para y se visualiza un informe de error. Por ejemplo, borre la línea 20, pulsando simplemente 20, luego ENTER. La línea 20 ha desaparecido. Pulse RUN y el ordenador le advertirá que no ha encontrado una variable que se utiliza en la línea 30.

LIST

La instrucción LIST visualiza el programa en la pantalla.

ENTER ejecuta lo mismo, entonces ¿por qué esta instrucción suplementaria?

Supongamos que usted ha introducido un programa de 40 líneas, numeradas de 10 a 400 en intervalos de 10 (esto es aconsejable, ya que usted podrá siempre intercalar, más tarde, otras líneas).

Pulsando ENTER, la pantalla muestra las últimas líneas del programa.

Pulsando LIST, la pantalla muestra las últimas líneas del programa. Pulsando LIST, la pantalla muestra el programa a partir de la primera línea del programa. Cuando la pantalla está llena, el ordenador visualiza: **scroll? ¿hay que efectuar desfile?**

Pulsando n, de no, el ordenador interrumpe el listado; pulsando cualquier otra tecla, el ordenador continúa el listado. Esto constituye una primera diferencia entre ENTER y LIST.

LIST x

Otra diferencia es que **la instrucción LIST admite un parámetro. Este parámetro x es el número a partir del cual se efectuará el listado.**

LIST 50 lista su programa, a partir de la línea 50 y, además, pone el indicador de línea editada > en la línea 50.

Esto es muy importante cuando usted debe modificar, por ejemplo, la línea 50. Si el indicador apunta a la línea 400, usted deberá desplazarlo a la línea 50, mediante sucesivas pulsaciones de ⇧ , lo que resulta bastante fastidioso. Teclee LIST 50 y podrá entonces, editar la línea 50 para proceder a realizar las correcciones.

NEW

Para introducir otro programa, no borre todas las líneas del programa antiguo, simplemente pulse **NEW.**

La instrucción NEW reinicializa el Spectrum y hace aparecer el mensaje copyright de nuevo.

Como no existe ninguna restricción en cuanto al número de caracteres por línea de programa, si usted confecciona una línea x de programa que llene 20 líneas de la pantalla, LIST x facilitará su visualización.

En resumen

- *Un programa exige la numeración de las líneas que lo componen.*
- *El indicador > apunta a la última línea editada.*
- *La edición y corrección de una línea, se efectúa con CAPS SHIFT y una de las teclas 5 a 8.*
- *La instrucción* **LIST** *facilita la edición de determinadas líneas y permite el listado de todo el programa de fragmentos.*
- *La instrucción* **NEW** *borra el programa y las variables.*

ATRIBUTOS

Hemos visto en el capítulo 2 que la pantalla tiene 768 posiciones PRINT. Cada una de estas posiciones tiene asignados unos atributos que le son propios. **Estos atributos son: INK, PAPER, FLASH, BRIGHT, OVER e INVERSE.**

Veamos sus acciones.

INK x

x es una cifra de 0 a 7. Representa uno de los 8 colores disponibles. El color escogido se asigna a los caracteres visualizados por la instrucción. Efectúe:

```
PRINT INK 4; "buenos días"
```

y la pantalla visualizará buenos días en color verde.

PAPER x

x es una cifra de 0 a 7 y representa uno de los 8 colores.
El color escogido se asigna al fondo de los caracteres visualizados por la instrucción. El color papel, junto con el color tinta, ofrece la manera de asignar 2 colores a cada posición PRINT.

```
PRINT PAPER 5;"yo soy el "
```

y la pantalla visualiza el texto sobre fondo cyan (azul claro).

FLASH x

x = 1 provoca el parpadeo del texto visualizado por la instrucción.
x = 0 suprime el parpadeo.

Introduzca:

```
PRINT FLASH 1;"spectrum"
```

y la palabra spectrum parpadeará en la pantalla.

BRIGHT x

x = 1 resalta el brillo del texto visualizado por la instrucción.
x = 0 da el brillo normal.
Introduzca:

```
PRINT BRIGTH 1;"a su disposición "
```

y el fondo del texto tendrá un tono mucho más claro.

Nota: Véase también el anexo 1 para todos estos atributos.

OVER x

x = 1 provoca la sobreimpresión exclusiva del texto visualizado por la instrucción.
x = 0 vuelve a la impresión normal.

Esta instrucción especial es bastante divertida de utilizar. Da resultados espectaculares y, cuando se domina bien, origina nuevos caracteres.
Como se trata de sobreimpresión, un texto debe ocupar el lugar en el que se desee visualizar, sobreimprimir, otro texto.

Cada carácter se compone de pequeños puntos llamados pixels.
Hay 64 –8 líneas de 8 pixels– para cada carácter. **Cuando un pixel sobreimprime un pixel ya presente, lo borra. Cuando un pixel sobreimprime un blanco, se visualiza.**

Introduzca sucesivamente:

```
PRINT OVER 1; AT 6,0; "control"
```

la pantalla visualizará la palabra: control

```
PRINT OVER 1; AT 6,0; "completo"
```

las letras c y o desaparecen, la letra final o se visualiza y entre las dos, se sobreimprimen las letras ntrol y mplet.

```
PRINT OVER 1' AT 1' AT 6,0 "completo"
```

y la palabra control reaparece.
Cuando se teclea OVER 1 dos veces seguidas, con la misma palabra, se anulan los dos mandatos y reaparece la primera palabra. Si, en el tercer mandato, usted hubiese introducido la palabra control, usted habría vuelto a encontrar la palabra completo.

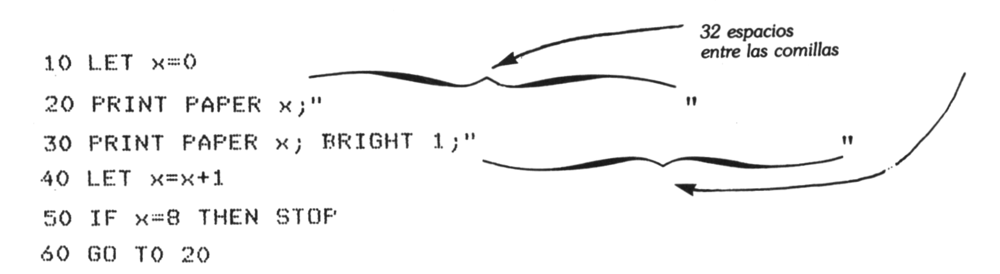
INVERSE x

x = 1 visualiza los caracteres en video invertido.
s = 0 los visualiza en video normal.
Efectúe:

```
PRINT INVERSE 1; "hola, amigo"
```

Visualización de colores

Introduzcamos un pequeño programa para visualizar los 8 colores disponibles, y estos mismos colores, con el brillo resaltado, de forma alternativa; obtendremos 16 matices.




```
10 LET x=0
20 PRINT PAPER x;"                                "
30 PRINT PAPER x; BRIGHT 1;"                                "
40 LET x=x+1
50 IF x=8 THEN STOP
60 GO TO 20
```

Nota: ponga 32 espacios entre las comillas de las líneas 20 a 30.

Este programa nos permite descubrir tres nuevas instrucciones.

IF... THEN...

Esta instrucción permite tomar una decisión, cuando se efectúa una comparación sobre una variable.

En el programa anterior, la comparación se produce sobre la variable x. Si x es igual a 8, se para el programa. Si x es distinto de 8, continúa con la línea siguiente: la línea 60. Cuando usted introduce la línea 50 en la zona de edición, ve que, después de la introducción de la palabra clave THEN, el cursor vuelve a ser [K]. Así pues, será necesario teclear una palabra clave que represente una instrucción. Las palabras claves que representan funciones, los caracteres, los signos, etc., no se admiten.

Esta manera de tratar la instrucción IF... THEN... proporciona una gran agilidad de utilización. Más adelante, veremos que el signo : (dos puntos) permite varias declaraciones en la misma línea de programa. Una comparación verdadera origina, de esta manera, varias acciones. Después de THEN, usted puede introducir IF (es una instrucción) y ejecutar varias comparaciones.
Borre las líneas 50 y 60 e introduzca ésta:

```
50 IF x<8 THEN GO TO 20
```

Obtendrá el mismo resultado y economizará una línea de programa.

GO TO x

Esta instrucción representa una salto incondicional. Esto significa: saltar a la línea que lleva el número x. Si x no existiera, el salto se efectuaría a la línea siguiente.
Si, en el programa anterior, usted efectúa

```
60 GO TO 15
```

el programa se ejecuta de la misma forma.

STOP

La instrucción STOP provoca la parada del programa.
El programa se para también automáticamente cuando se han ejecutado todas sus líneas. **La instrucción STOP** no es necesaria al final de un programa, **sirve para parar el programa en curso de ejecución.**

Nota: en los anexos, usted encontrará todas las instrucciones y funciones que comprende su Spectrum, con explicaciones y ejemplos de funcionamiento. En los capítulos posteriores, no volveremos a repetir estas explicaciones, salvo para precisar ciertas particularidades.

Atributos

Los 6 atributos, descritos más arriba, se ponen como artículo en una declaración PRINT, y sólo afectan a esta declaración.

Todas las instrucciones que visualizan algo en la pantalla, pueden contener artículos de color, que sólo afectan localmente a la pantalla.

Por otra parte, los 6 atributos de arriba pueden utilizarse como instrucciones. En este momento, se encontrarán afectadas todas las posiciones PRINT de la pantalla.

En el programa anterior, efectúe:

```
5 FLASH 1
```

y se producirá el parpadeo en toda la pantalla.

En resumen

- *Los* **atributos** *se utilizan localmente, como artículos de color u otros, en el interior de las instrucciones de visualización, o globalmente, de manera permanente, en las declaraciones.*
- *La instrucción* **IF... THEN...** *permite comprobar una condición y orientar la continuación del programa, según esta condición sea verdadera o falsa.*
- *La instrucción* **GO TO** *es un salto incondicional.*

INTRODUCCIÓN DE VARIABLES POR PROGRAMACIÓN

INPUT c; "mensaje"; x

La instrucción INPUT, en una declaración, detiene el desarrollo del programa cuando éste llega al número de línea donde se encuentra.

Un cursor parpadeante le solicita la introducción de una variable numérica; **si el cursor parpadeante se encuentra encerrado entre comillas, usted introduce entonces una variable de cadena.** Para precisar la variable solicitada, usted añadirá un mensaje. Para adornar o para llamar la atención, se colocan en la declaración uno o varios artículos de color. El mensaje y el cursor aparecen en la zona de edición.

Cálculo de la TVA

Introduzca el programa siguiente:

```
  10 PRINT AT 0,8;"Cálculo de la
TVA",,,
  20 INPUT INK 5; INVERSE 1;"Qué
tipo? (introduzca 25  por  25 por
ciento, 12 por 12 por ciento, etc
...)",x
  30 INPUT BRIGHT 1;"¿Qué  número
?",y
  40 PRINT "La TVA sobre ";y;" al
tipo  de ";x;"/100;" es: ";  x*y
/100
  50 GO TO 30
```

Nota: los trazos que subrayan indican un espacio (no ponga ningún signo).

El programa se presenta tal como se visualizará en la pantalla, a fin de explicar determinadas particularidades.

• **Línea 10:** Al final de la línea, hay 3 comas: la primera desplaza la posición PRINT al principio de la línea siguiente, la segunda al medio de la línea siguiente, y la tercera al principio de la segunda línea siguiente. El resultado habría sido el mismo, si se hubieran colocado 2 apóstrofos.

• **Línea 20:** el primer artículo de color selecciona el color cyan para los caracteres que se visualizarán; el segundo artículo de color provoca la inversión, así pues, en vez de obtener caracteres azules sobre fondo blanco, se obtendrán caracteres blancos sobre fondo azul.
Se visualiza el mensaje, entre comillas, señalando que, ya que cada línea de la pantalla contiene 32 caracteres, la Q de Que, la p de por y la e de etc., se alinean en la misma columna. Estas 3 letras inician 3 líneas.
Después del mensaje, hemos colocado una coma para que el cursor parpadeante aparezca en medio de la línea.
x es el nombre de la variable a la que se asignará el valor que usted introduzca.

• **Línea 40:** en determinados emplazamientos, donde el espacio es necesario, hemos colocado un trazo para llamar su atención; véase la nota de arriba.

• **Línea 50:** para volver a la línea 30, a fin de calcular la TVA, a partir del número que usted introduzca. Las líneas 30, 40 y 50 constituyen un circuito que no para nunca. El ordenador le pedirá sin cesar: ¿Qué número?

Para parar este programa, engañe al ordenador. Cuando le pida ¿Qué número?, introduzca cualquier letra. El ordenador se parará y visualizará un mensaje de error: variable no encontrada. Para introducir otro tipo de TVA, pare el programa y después, vuelva a arrancarlo. O bien, después de pararlo, teclee los mandatos:

```
LET x=18
GO TO 30
```

si usted desea un tipo del 18 por ciento.

Variables

En los nombres de las variables, el ordenador no diferencia entre una minúscula y una mayúscula. En cambio, **tiene en cuenta todas las letras del nombre de la variable.** LIMÓN es la misma variable que limón, pero es distinta de limones. Efectúe

```
LET limón=5
PRINT limón                    5
PRINT LIMON                    5
PRINT limones          variable no encontrada
```

No existe ninguna restricción en el nombre de las variables. Ya que las palabras compuestas, instrucciones y funciones, se codifican en un solo octeto, podemos tomarlas como nombres de variables.

```
LET SIN=2
PRINT sin                      2
```

La palabra SIN se introduce, evidentemente, con tres letras.

VARIABLES DEL SISTEMA

POKE, PEEK, célula de memoria

Efectúe el mandato siguiente:

```
POKE 23609,30
```

el cual le aconsejamos que realice cada vez que ponga en marcha su ordenador.

Usted necesita variables para construir su programa, lo mismo que el ordenador para funcionar. Las que utiliza el ordenador se llaman "variables del sistema". Para realizar las distintas órdenes que usted le da, el ordenador extrae el valor de las variables del sistema afectadas por tales órdenes, las utiliza, las maneja y, después, las almacena de nuevo en la memoria para su posterior uso.

Para conocer su contenido se utiliza la función PEEK; y para modificar su contenido se utiliza la instrucción POKE. De esta manera, el mandato de arriba ha situado 30 en la variable del sistema denominada PIP (véase el manual Sinclair), cuya dirección es 23609. En la inicialización, su valor es 0. Situando 30, el clic producido al pulsar cada tecla es más audible, y le permite conocer si se ha tenido en cuenta la pulsación de una tecla, sin estar obligado a mirar la pantalla. Usted puede introducir un valor de 0 a 255. El ordenador rechazará un valor más grande de 255.

Cada célula de la memoria del ordenador contiene un octeto. El octeto está formado por 8 cifras binarias. Si convertimos el número representado por estas 8 cifras binarias en un número decimal, cada célula contendrá un número de 0 a 255. Si deseamos manejar un número más grande de 255, necesitaremos una célula de memoria suplementaria para representarlo.
El número 256 necesitará 2 células de memoria: la primera contendrá 0 y la segunda 1.
El número 255 está contenido en una célula: | 255 |
El número 256 está contenido en dos células: | 0 | 1 |
que se leen: 0 veces 1 + 1 vez 256
Así, primero el octeto de menos peso, y después el de mayor peso.

Un neófito encontrará extraña esta manera de proceder, pero la lógica del ordenador lo requiere así. Solamente existe una excepción a esta regla en el Spectrum.

Cada célula de memoria lleva una dirección. Las variables del sistema utilizadas por el Spectrum poseen las direcciones 23552 a 23733.

Para conocer el contenido, se utiliza la función PEEK. Por ejemplo, la variable del sistema PIP, de dirección 23609:

```
PRINT PEEK 23609
```

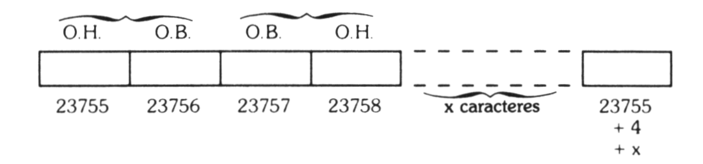
visualiza 30, si usted ha efectuado el mandato del principio del capítulo.

Una variable del sistema, que contenga un número más grande de 255, necesitará 2 células de memoria. De esta manera, la variable del sistema PROG, que contiene la dirección de

inicio del programa Basic, está ubicada en las direcciones 23635 y 23636. Para obtener el contenido de PROG, efectúe:

```
PRINT PEEK 23635 + 256*PEEK 23636
```

el cual visualiza 23755: es el primer octeto de la zona de programa de la memoria.
He aquí la construcción de una línea de programa:



Los dos primeros octetos de la zona de programa, 23755 y 23756, contienen el número de la primera línea de programa. Como las líneas de programa deben estar numeradas por un número entero, comprendido entre 1 y 9999, se necesitan 2 octetos, y aquí tenemos la única excepción del sistema: primero el octeto alto, y después, el octeto bajo. Los dos octetos siguientes, 23757 y 23758, contienen el número de caracteres x que componen la línea Basic; por consiguiente, una línea Basic puede contener más de 255 caracteres. La dirección del primer octeto de la línea siguiente será 23755 + 4 + x.
Note que x es el número de caracteres, tal como usted los introduce, más el carácter ENTER, más tantas veces 6 octetos como números a convertir en coma flotante.

Con estos elementos, construyamos un programa, para renumerar las líneas Basic de un programa en curso de elaboración.

Programa de renumeración

```
9900 REM  renumeración
9901 LET nuevo=10
9902 LET inc=10
9910 LET adr=PEEK 23635 + 256*PEEK
9920 LET no=256*PEEK adr+PEEK (adr+1)
9930 LET long=PEEK (adr+2)+256*PEEK (adr+3)
9940 IF no=9900 THEN STOP
9950 LET octH=INT (nuevo/256)
9960 POKE adr,octH : POKE adr+1,nuevo-256*octH
9970 LET nuevo=nuevo+inc
9980 LET adr=adr+4+long
9990 GO TO 9920
```

Las variables que se utilizan son:

nuevo:

es el nuevo número de línea que quiere asignar a la primera línea de su programa. Usted cambiará esta variable, si desea renumerar a partir de 100, por ejemplo, para introducir líneas por debajo, ponga 100.

inc:

es el incremento que usted desea añadir, para obtener el número de línea siguiente. Cámbielo si fuera necesario.

adr:

es la dirección de inicio de la zona de programa.

no:

es el número de línea antes de la renumeración. El cálculo de esta variable es necesario para ejecutar la línea 9940, a fin de parar la renumeración cuando se llegue a la línea 9900.

long:

es la longitud de la línea Basic.

oct H:

es el octeto alto del nuevo número de línea.

Salve este programa de utilidad en cassette.
Más tarde, cuando usted programe, **cárguelo con el mandato MERGE** y, todas las veces que desee renumerar, teclee el mandato RUN 9900.

En el anexo 5, se ofrecen numerosos detalles sobre las variables del sistema.

DEFINICIÓN DE FUNCIONES

DEF FN

Esta potente instrucción le permite definir sus propias funciones, cualesquiera que sean. Si usted encuentra que las instrucciones y funciones previstas por Sinclair para su Spectrum son insuficientes, cree otras.

Generalmente, **se utiliza DEF FN para reemplazar una línea Basic que se utiliza a menudo.** Así se evita el volver a escribirla todas las veces que se presente su utilización.

Definición de una función numérica

En el programa anterior, las líneas 9910 y 9930 tienen la misma función: se trata de extraer un número contenido en dos octetos sucesivos de la memoria. Transformémoslos creando una función personal.

```
DEF FN x(y)=PEEK y+256*PEEK (y+1)
```

La función definida se denomina x y su argumento y es ficticio.
Se reemplazará por un número o por una expresión.
Las líneas 9910 y 9930 se convierten:

```
LET adr=FN x(23635)
LET long=FN x(adr+2)
```

Todo esto está muy bien, pero la línea 9920, que calcula una función parecida, pero con inversión de los octetos altos y bajos, ¿necesita una nueva DEF FN?... Pues bien, no. La agilidad de DEF FN es tal que, añadiéndole un segundo argumento, resolvemos el problema.

Transformemos las siguientes líneas del programa anterior:

```
9903 DEF FN x(y,z)=PEEK y+256*PEEK (y+z)
9910 LET adr=FN x(23635,1)
9920 LET no=FN x(adr+1,-1)
9930 LET long=FN x(adr+2,1)
```

Nota: Si usted desea renumerar su programa a partir, por ejemplo, de la línea 40, añada la línea:

```
9945 IF no<40 THEN GO TO 9980
```

y cambie, igualmente y obligatoriamente, la línea 9901. Usted pone aquí el número que quiere que tome la línea 40.

Igualmente, si usted desea evitar la renumeración de un programa que empiece en 1000, cambie la línea 9940:

```
9940 IF no=1000 THEN STOP
```

Definición de una función de cadena

La función definida tiene un nombre de una letra, como para la numérica, seguida del signo dólar.

Escribamos, por ejemplo, una función para convertir un número decimal de 0 a 15 en un número hexadecimal de 0 a F.

```
DEF FN x$(y)=CHR$ (48+y+7*(y>9))
```

Para ir de 0 a 9, la función proporciona el carácter de código ASCII 48 + y, y para ir de 10 a 15, la función proporciona el carácter de código ASCII 48 + y + 7. En el primer caso, el 2.° paréntesis es falso, así pues, igual a 0; y en el segundo caso, es verdadero, así pues, igual a 1.

Programa para convertir un número decimal en hexadecimal

```
 10 REM  conversión dec/hexa
 20 INPUT "Introduzca un número decimal",d
 30 IF d>255 THEN GO TO 90
 40 LET h=INT (d/16)
 50 LET l=d-16*h
 60 LET d$=FN x$(h)+FN x$(l)
 70 PRINT d$
 80 GO TO 20
 90 IF d>65535 THEN PRINT "Su número es demasiado grande" : STOP
100 LET hh=INT (d/4096)
110 LET lh=INT ((d-4096*hh)/256)
120 LET d$=FN x$(hh)+FN x$(lh)
130 PRINT d$;
140 LET d=d-4096*hh-256*lh
150 GO TO 40
999 DEF FN x$(y)=CHR$ (48+y+7*(y>9))
```

La línea que define una función puede estar en cualquier lugar del programa. Tenga la costumbre de ponerla al principio o al final del programa.

Dado que FN x$(y) es una cadena, se utiliza el signo + en las líneas 60 y 120 como signo de encadenamiento.

Las variables utilizadas son:

d :

número decimal que usted introduce.
Para d hasta 255, el programa proporciona 2 cifras hexadecimales.
Para d, de 256 a 65535, el programa proporciona 4.
Para d mayor que 65535, el programa se para. Ésta es una manera de parar el programa, que bifurca constantemente a la línea 20.

h :

parte alta del octeto bajo

l :

parte baja del octeto bajo. Usted podría utilizar L para evitar la confusión con la cifra 1.

d$:

cadena de 2 cifras hexadecimales

hh :

parte alta del octeto alto

lh :

parte baja del octeto alto. También aquí, puede ser preferible LH.

Tenga en cuenta que, cuando hay que visualizar 4 cifras hexadecimales, la línea 130 termina en un ; (punto y coma), lo que pone juntas las 2 cifras hexadecimales siguientes de la línea 70. Esta última línea provoca un paso a la línea siguiente de la posición PRINT. Para visualizar igualmente el número decimal, añada la línea:

```
25 PRINT d,
```

CIRCUITOS (BUCLES)

FOR... TO... STEP... NEXT

Hemos visto anteriormente la utilización del GO TO para saltar a una línea del programa. Si esta línea de programa precede al número de la línea que contiene el GO TO, se producirá un circuito permanente.

Para visualizar los números naturales, efectúe:

```
10 LET a=1
20 PRINT a
30 LET a=a+1
40 GO TO 20
```

RUN visualiza los números 1 a 22 y el ordenador solicita **scroll?**
Pulse una tecla y el ordenador visualizará los números 23 a 44.
Si usted pulsa N o SPACE, el programa se para.

Para visualizar 4 números por línea, transforme la línea 20, de la manera siguiente:

```
20 PRINT a;TAB 8*a;
```

Si usted desea limitarse a los 50 primeros números naturales, añada la línea 35 y su programa se convertirá:

```
10 LET a=1
20 PRINT a;TAB 8*a;
30 LET a=a+1
35 IF a=51 THEN STOP
40 GO TO 20
```

La utilización de un circuito automático simplifica este programa:

```
10 FOR a=1 TO 50
20 PRINT a;TAB 8*a;
30 NEXT a
```

Los nombres de las variables que controlan los circuitos solamente contienen una letra. De esta forma, usted dispone de 26 variables de control de circuitos. Como un circuito puede contener otro circuito, y este lo mismo, etc..., esto permite numerosos enlaces posibles. Cuando un circuito ha terminado, usted puede volver a utilizar esta variable para controlar el circuito siguiente.
En el interior de un circuito, usted puede utilizar la variable de control, pero no puede modificarla.

10 FOR a = 1 :

Usted declara un circuito cuya variable de control es a y cuyo valor inicial es 1.

TO 50 :

Usted declara el valor final, o límite de la variable de control a. Después de la declaración de un circuito, el ordenador ejecuta lo que sigue, hasta que encuentra la instrucción NEXT a

30 NEXT a :

el ordenador incrementa la variable de control a, después verifica si este nuevo valor de a supera el valor límite fijado; si lo supera, el ordenador continuará con la instrucción siguiente; si no, volverá a la declaración que sigue a aquélla, en la que usted ha declarado el circuito.

```
FOR a = 1 TO 50 STEP 7
```

Si, en la declaración de un circuito, usted especifica el paso de incremento de la variable de control, NEXT a añade este paso a la variable, después verifica la superación eventual del límite, luego efectúa un salto.

En el caso de arriba, el circuito se efectuará 8 veces, con los valores de a siguientes: 1, 8, 15, 22, 29, 36, 43, 50.

Después del desarrollo del programa, tecleando el mandato PRINT a, el ordenador visualizará 57.

Como ve, de todas maneras, el ordenador añade el incremento a la variable de control, incluso si a continuación decide salir del circuito.

Tabla de senos de un ángulo expresado en grados

```
10 PRINT TAB 8;"TABLA DE SENOS"
20 PRINT
30 FOR a=0 TO 90 STEP 5
40 LET b=a*PI/180
50 PRINT "senos ";a;" = ";TAB 20;SIN b
60 NEXT a
```

Respete los espacios de la línea 50.

Esta tabla da los senos de 0 a 90 grados, de 5 en 5 grados. Usted puede visualizar una tabla de 40 a 42 grados, de 0.1 en 0.1 grados, o una tabla de cosenos, o de tangentes, etc.

Si desea los logaritmos decimales, en lugar de los neperianos, utilice este programa:

Tabla de logaritmos decimales

```
10 PRINT "TABLA DE LOGARITMOS DECIMALES"
20 PRINT
30 PRINT "Nota: LOG es el logaritmo de base 10"
40 PRINT
50 LET x=10
60 FOR n=x TO 10*x STEP x
70 PRINT "LOG ";n;" = ";TAB 10;LN n/LN 10
80 NEXT n
```

Usted obtiene los logaritmos de los números 10 a 100. Si los quiere de 1 a 10, teclee:

```
50 LET x=1
```

Modifique la línea 60 para obtener otras horquillas y otro incremento.

Radios de ruedas en diagonal

```
10 FOR n=17 TO 153 STEP 17
20 LET m=25*n/17
30 FOR j=0 TO 2*PI STEP PI/6
40 PLOT m,n
50 DRAW 15*COS j,15*SIN j
60 NEXT j
70 NEXT n
```

Obtendrá usted 9 ruedas dispuestas en diagonal. Para cada rueda, se visualizan 12 radios. (Las instrucciones PLOT y DRAW se detallarán más adelante).

Este programa ilustra dos circuitos enlazados, lo que hace igualmente el programa siguiente.

Radios de rueda

```
10 PRINT AT 11,16; "■" : REM modalidad G tecla 1
20 FOR n=2 TO 10 STEP 2
30 FOR j=0 TO 2*PI STEP PI/16
40 PRINT AT 11+n*SIN j,16+n*COS j;"■"
50 NEXT j
60 NEXT n
```

Usted obtendrá 12 radios de una rueda, (n.sin j, n.cos j) son las coordenadas de un círculo, de radio n y de centro (11, 16).

Los parámetros de un PRINT AT pueden ser expresiones, y lo hemos aprovechado. Igualmente, es la ocasión de dar a conocer varios circuitos y la utilización de sus variables de control para modificar posiciones PRINT.

En los anexos 1 y 2 se proporcionan numerosos ejemplos y detalles suplementarios sobre las instrucciones y las funciones.

CADENAS DE CARACTERES

CHR$ CODE LEN SCREEN$ (x,y) STR$ VAL + TO

CHR$ x proporciona el carácter cuyo código es x

```
PRINT CHR$ 65
```

visualiza A. 65 es el código ASCII de A.

```
PRINT CODE "A"
```

visualiza 65.

CODE proporciona el código ASCII del carácter A.

Estas dos funciones son inversas la una de la otra.

```
PRINT CHR$ CODE "A"
```

visualiza A.
CODE funciona igualmente con una cadena, pero solamente da el código del primer carácter de la cadena.

```
PRINT CODE "ALBARICOQUE"
```

visualiza 65. Esto permite las manipulaciones alfabéticas.

LEN a$ proporciona la longitud de la cadena a$.

SCREEN$ (x,y) da el carácter situado en las coordenadas (x,y).

En el anexo 2, encontrará usted un ejemplo de estas dos funciones dentro de la sección SCREEN$.

STR$ x transforma el número o la expresión numérica x en cadena.

VAL "f" proporciona el valor de f, siempre que éste sea una variable numérica o una expresión numérica.

Se utiliza a menudo VAL para economizar octetos de memoria.
Algunos programas largos necesitan a veces de una cierta astucia para economizar memoria.

En todas las líneas de programa donde aparece un número, el ordenador hace que este número vaya seguido del carácter 14, seguido de 5 octetos, que representan este número en coma flotante, o sea 6 octetos en total. Estos 6 octetos están presentes en la zona de programa, pero no aparecen en el listado del programa. Por ejemplo, haga:

```
10 FOR n=NOT PI TO VAL "4"
20 PRINT AT 5*n,5*n; "Hola amigo"
30 NEXT n
```

Usted verá algunas veces, estudiando programas publicados en revistas, el mismo tiempo que la línea 10 de arriba. En ésta, NOT PI reemplaza a 0 (2 octetos en lugar de 7), y VAL "4" reemplaza a 4 (4 octetos en lugar de 7).
Es la misma línea que:

```
10 FOR n=0 TO 4
```

pero economiza 8 octetos de memoria.

+ es el signo de encadenamiento de cadenas.

```
10 LET a$="BUENOS"
20 LET b$=" DIAS"
30 PRINT a$+b$
```

visualiza BUENOS DÍAS.

En cambio esto:

```
10 LET a$="4"
20 LET b$="5"
30 PRINT a$+b$
40 PRINT VAL a$+VAL b$
```

visualiza

```
45

 9
```

En la línea 40, el signo + vuelve a ser una operación aritmética.

TO es el separador de cadena

Sirve “para todo”; sustituye ventajosamente las distintas instrucciones de truncaje de cadenas, encontradas en los Basic más antiguos, tales como LEFT$, MI$, RIGHT$, etc.
Efectúe los mandatos:

```
LET x$ ="BUENOS DIAS"
PRINT x$(TO 7)                  visualiza BUENOS
PRINT x$(8 TO)                  visualiza DIAS
PRINT x$(2 TO 3)+xS(6 TO 6)     visualiza UES
PRINT x$ (TO)                   visualiza BUENOS DIAS
```

TO debe estar situado entre las posiciones de los caracteres de la cadena; pero cuando no hay nada a la izquierda de TO, significa desde el principio; y cuando no hay nada a la derecha de TO, significa hasta el final.

Los delimitadores pueden ser expresiones:

```
PRINT x$(2*2 TO 2+3)             visualiza NO
```

El ZX Spectrum posee un evaluador de expresiones muy potente, que puede utilizarse casi en todas partes.

Anagrama – La palabra más larga

He aquí un programa que sustituye simplemente dos letras, la una por la otra, en una palabra; y esto a cada pasada.

A partir de una palabra, que usted introduce, obtiene sucesivamente palabras, en las que se han invertido 2 letras.

Usted puede introducir 9 letras y jugar a la palabra más larga.

```
 10 REM   Anagrama
 20 INPUT "Introduzca una palabra ",a$
 30 LET a=LEN a$
 40 LET b=1+INT (RND*a)
 50 LET c=1+INT (RND*a)
 60 LET b$=a$(b)
 70 LET a$(b)=a$(c)
 80 LET a$(c)=b$
 90 PRINT a$
100 GO TO 40
```

Introduzca ALBARI, COQUE, etc., y observe el desarrollo.
Si usted introduce CAELMOPRT, hará falta tiempo para que el ordenador visualice COMPLETAR.

Cambiemos ligeramente este programa y añadamos:

```
22 INPUT "Introduzca el principio de la palabra ",x$
```

modifiquemos:

```
90 PRINT x$+a$
```

Usted cree que una palabra que empiece por COM es posible, haga:

AELPRT como 1.ª introducción, y
COM como 2.ª introducción

el ordenador encontrará más rápidamente COMPLETAR.

Usted imagina que la palabra termina por TAR; añadamos estas líneas:

```
24 INPUT "Introduzca el final de la palabra ",y$
90 PRINT x$+a$+y$
```

y haga

CELMOP	como 1.ª introducción
nada	como 2.ª introducción
TAR	como 3.ª introducción

o bien

ELP	como 1.ª introducción
COM	como 2.ª introducción
TAR	como 3.ª introducción

El programa siguiente contiene una matriz de caracteres, dimensionada en 10 cadenas de 10 caracteres como máximo, con la instrucción DIM.

Ordenación alfabética

```
 10 REM  orden alfabético
 20 DIM a$(10,10)
 30 FOR n=1 TO 10
 40 INPUT "Introduzca una palabra ",a$(n)
 50 PRINT a$(n)
 60 NEXT n
 70 FOR m=1 TO 10
 80 FOR n=2 TO 10
 90 IF a$(n) a$(n-1) THEN LET b$=a$(n)
: LET as(n)=as(n-1) : LET as(n-1)=b$
100 NEXT n
110 NEXT m
200 FOR n=1 TO 10
210 PRINT AT n-1,16;a$(n)
220 NEXT n
```

Introduzca sucesivamente: LA, QUE, TRISTE, NEGRURA, EVITAR, DEVUELVA, CONCIENCIA, ADELANTE, BUENA, TENER

- El circuito 30 a 60 pide 10 palabras, que usted introduce; y las visualiza.
- El circuito 80 a 100 compara cada palabra con la anterior y, si es más pequeña, toma su sitio.
- El circuito 70 a 110 hace ejecutar 10 veces el circuito anterior, para que la 10.ª palabra introducida pueda situarse en 1.ª posición, si fuera necesario.
- El circuito 200 a 220 visualiza las palabras ordenadas, al lado de las palabras introducidas.
- En la línea 90, lo que sigue a THEN es de la forma:

$$x = b$$
$$b = c$$
$$c = x$$

para invertir los valores de b y c.

Esta forma ha sido igualmente utilizada en el programa Anagrama.

GRÁFICOS DEFINIDOS POR EL USUARIO

BIN

Los GDU (o UDG, en inglés) son caracteres nuevos que usted crea, según su criterio, en una rejilla de 64 cuadrados. Estos caracteres nuevos representan, según su creación, símbolos matemáticos, letras griegas, pequeños dibujos, etc.

Cuando su Spectrum se pone en marcha, usted puede definir 21. Puede aumentar el número de GDU por series de 21 y de esta manera definir 42, 63, 84, etc. nuevos gráficos. El medio empleado está descrito en el anexo 5. Igualmente, en el anexo 5 damos un medio para volver a definir los 96 caracteres visualizables. Todo esto le da a usted una gran variedad de caracteres y gráficos.

Cada posición PRINT de la pantalla es una rejilla de 8 × 8 casillas. **Cada casilla se llama pixel.** Cuando este pixel está a 1, la correspondiente casilla se visualiza en color tinta, y cuando el pixel está a 0, en color papel.

Supongamos un rejilla de 8 veces 8 casillas.

línea									binario	decimal
1				●	●				11000	24
2			●			●			100100	36
3		●					●		100010	66
4	●							●	10000001	129
5	●							●	10000001	129
6		●					●		1000010	66
7			●			●			100100	36
8				●	●				11000	24

En esta rejilla, usted dibuja el carácter deseado, ennegreciendo una casilla o poniendo un punto. Así, dispone de 8 líneas diferentes, que debe introducir en la zona UDG. Ésta empieza en 32600 en el Spectrum de 16 k, y en 65368 en el de 48 k.

Usted tiene la posibilidad, en el Spectrum, de introducir un número en forma binaria, utilizando BIN, o en forma decimal. En ambas numeraciones, los ceros situados a la izquierda de la primera cifra significativa, pueden omitirse.

La forma binaria es fácil de utilizar; usted coloca 1 en los lugares que ha ennegrecido o punteado, y 0 en los otros.

```
POKE 32600,BIN 11000
```

en el mismo mandato que

```
POKE 32600,24
```

Aquí dispone de un medio para convertir un número binario (de no más de 8 cifras 0 y/o 1) en número decimal.

```
PRINT BIN 11000
```

Introduzca a continuación los 7 octetos siguientes:

```
POKE 32601,BIN 100100
POKE 32602,BIN 1000010
POKE 32603,BIN 10000001
POKE 32604,BIN 10000001
POKE 32605,BIN 1000010
POKE 32606,BIN 100100
POKE 32607,BIN 11000
```

Cuando los 8 octetos se introducen en memoria,

```
PRINT CHR$ 144
```

visualiza un cuadrado en punta. Y el programa:

```
10 PRINT CHR$ 144;
20 GO TO 10
```

llena la pantalla de cuadrados en punta. ¿Verdad que es bonito?.

Los caracteres de código 144 a 164, son los caracteres UDG; véase el anexo 3.

Como es pesado introducir todos estos octetos, veamos otra forma de proceder: éste es el objeto del capítulo siguiente.

INTRODUCCIÓN DE PARÁMETROS POR PROGRAMA

DATA READ RESTORE

La instrucción DATA almacena los parámetros que usted necesita.
Usted continúa DATA con uno o varios paráämeros, escritos en decimal, o en binario con BIN. Las cadenas y las expresiones son igualmente parámetros válidos de la instrucción DATA.

La instrucción READ extrae sucesivamente cada parámetro y le asigna una variable.

En cuanto a RESTORE, escoge la línea de DATA que usted desea utilizar.

Para el cuadrado en punta del capítulo anterior, se introduce:

```
10 DATA 24,36,66,129,129,66,36,24
```

o bien, en binario:

```
  10 DATA BIN 11000,BIN 100100,BIN 1000010,BIN 10000001,
BIN 10000001,BIN 1000010,BIN 100100,BIN 11000
```

y a continuación,

```
20 FOR n=32600 TO 32607 : REM 65368 a 65375 para un 48 k
30 READ a
40 POKE n,a
50 NEXT n
```

y, para la visualización,

```
60 PRINT CHR$ 144;
70 GO TO 60
```

Con la ayuda de RESTORE, podemos desplazar una pequeña línea vertical, pixel a pixel.

Desplazamiento por pixel

```
 10 DATA 128
 11 DATA 64
 12 DATA 32
 13 DATA 16
 14 DATA 8
 15 DATA 4
 16 DATA 2
 17 DATA 1
490 FOR x=10 TO 17
500 FOR n=65368 TO 65375 : REM 32600 a 32607 para un 16 k
510 RESTORE x : READ a
520 POKE n,a
530 NEXT n
540 PRINT AT 0,0;CHR$ 144
550 NEXT x
```

El circuito 500 a 530 carga 8 octetos en el carácter UDG de código 144. RESTORE 10 obliga a READ a leer 8 veces el dato de la línea 10. Después, la línea 540 visualiza el carácter 144. En el siguiente valor de x, RESTORE 11 obliga a READ a leer 8 veces el dato de la línea 11. La línea 540 visualiza el nuevo carácter 144 en el mismo sitio, que siempre es una línea vertical, pero desplazada un pixel hacia la derecha. Y así sucesivamente.

Si usted desea la visualización punto por punto, suprima la línea 540 e introduzca la línea:

```
525 PRINT AT 0,0;CHR$ 144
```

y la línea vertical se desplazará serpenteando. Es divertido, ¿no?

Para hacer desfilar esta pequeña línea vertical de izquierda a derecha de la pantalla, añada las líneas:

```
480 FOR y=0 TO 30
560 NEXT y
```

y cambie la línea 540, de esta manera:

```
540 PRINT AT 0,y;CHR$ 32;CHR$ 144
```

en la que, AT 0,y; para hacer avanzar la posición PRINT, y CHR$ 32; (visualiza un espacio) para borrar el carácter cuando se cambia de posición PRINT.

En este caso tan preciso, por el hecho de que los datos son sucesivamente la mitad del dato precedente, puede prescindirse de las instrucciones DATA, READ y RESTORE.

```
470 LEF a=128
480 FOR y=0 TO 30
490 FOR x=1 TO 8
500 FOR n=65368 TO 65375
520 POKE n,a
530 NEXT n
540 PRINT AT 0,y;CHR$ 32;CHR$ 144
550 LET a=a/2
560 NEXT x
570 LET a=128
580 NEXT y
```

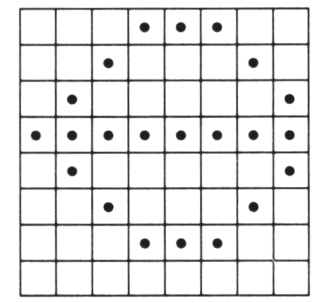
Antes de probar este programa, borre el programa anterior tecleando NEW.

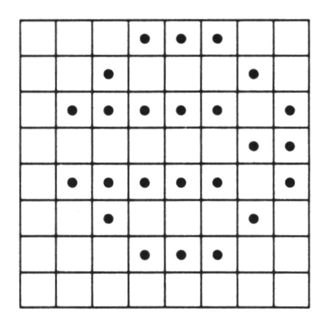
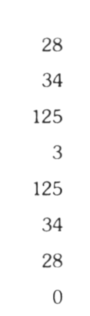
Para desplazar un carácter cuya matriz no está constituida por 8 octetos idénticos, habrá que manipular cada uno de estos octetos separadamente, y utilizar al menos dos caracteres UDG. Todo esto haría crecer el programa y la velocidad de ejecución sería lenta. **Es preferible utilizar la instrucción PLOT, que permite la animación de caracteres o de figuras gráficas con rapidez y sin hacer crecer el programa inútilmente.**

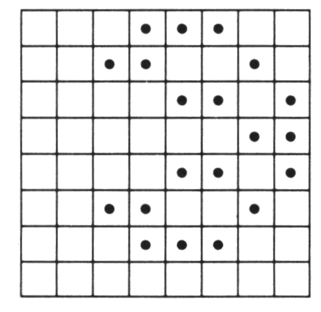
Sin embargo, es posible la animación en una sola posición PRINT, utilizando varios caracteres UDG.

La enzima glotona

Dibujemos varias enzimas en distintas posiciones:





28
50
13
3
13
50
28
0

			•	•	•			28
				•		•		10
					•		•	5
						•	•	3
					•		•	5
				•		•		10
			•	•	•			28
								0

Pongámoslas en 4 caracteres UDG:

```
 10 DATA 28,34,65,255,65,34,28,0
 20 DATA 28,34,125,3,125,34,28,0
 30 DATA 28,50,13,3,13,50,28,0
 40 DATA 28,10,5,3,5,10,28,0
 50 FOR n=65368 TO 65399
:   REM de 32600 a 32631 para un 16 K
 60 READ a : POKE n,a
 70 NEXT n
```

después, realicemos la animación con la ayuda de dos circuitos:

```
 80 LET x=10
 90 LET i=0
500 FOR n=0 TO 3
510 PRINT AT 10,16;CHR$ (144+n)
520 PAUSE x
530 NEXT n
540 FOR n=2 TO 0 STEP - 1
550 PRINT AT 10,16;CHR$ (144+n)
560 PAUSE x
570 NEXT n
580 GO TO 500
```

Mire la glotona.

Hagamos variar la velocidad de sus mandíbulas, añadamos:

```
 580 IF i=0 THEN LET x=x-1 : IF x=0
THEN LET i=1
 590 IF i=1 THEN LET x=x+1 : IF x=10
THEN LET i=0
 600 GO TO 500
```

¡A qué velocidad traga plancton esta enzima!

En las líneas 580 y 590 hemos utilizado un indicador i para hacer variar la velocidad de las mandíbulas, disminuyendo o aumentando x, según i sea igual a 0 o a 1.

La utilización de varias instrucciones en una sola línea hace que el programa sea más compacto y legible. La utilización de dos circuitos, uno para aumentar x, el otro para disminuir x, y la orientación hacia uno de los dos circuitos, habría necesitado más líneas de programa.

CAPTACIÓN DE CARACTERES

INKEY$

Hemos visto la introducción de datos por programa, mediante la instrucción INPUT, por una parte, y las instrucciones DATA y READ, por otra. **La función INKEY$ ofrece un medio de introducir un carácter sin interrumpir el desarrollo del programa.**

Desplazamiento de un objeto

Desplacemos la letra o en diagonal sobre la pantalla, evitando que salga de los límites de la misma.

```
 10 LET x=16 : LET y=11
 20 LET dx=1 : LET dy=1
 30 PRINT AT y,x;" " : REM   1 espacio
 40 LET x=x+dx : LET y=y+dy
 50 PRINT AT y,x;"O"
 60 IF x>30 THEN LET dx=-1
 70 IF x<1 THEN LET dx=1
 80 IF y>20 THEN LET dy=-1
 90 IF y<1 THEN LET dy=1
100 GO TO 30
```

La pelota se desplaza rebotando en los bordes.

Para tener la continuidad de visualización, el borrado de la pelota debe preceder inmediatamente a la visualización de la pelota en su posición siguiente. En este caso, solamente la línea 40 separa las líneas de borrado y de visualización; esta línea cambia las coordenadas de la pelota.

Desplazamiento de una raqueta

```
 210 LET t=9 : LET dt=0
 220 PRINT AT t,1;" ";AT t+1,1;" ";
AT t+2,1; " " : REM un espacio cada vez
 230 LET t=t+dt
 240 PRINT At t,1;CHR$ 124;AT t+1,1;
CHR$ 124;AT t+2,1;CHR$ 124
 250 LET dt=0
 260 IF INKEY$="6" AND t<19 THEN LET dt=1
 270 IF INKEY$="7" AND t>0 THEN LET dt=-1
 280 GO TO 220
```

• La raqueta está representada por tres caracteres 124 superpuestos. La línea 220 borra la raqueta, y la línea 240 la visualiza en la columna 1 y, en el arranque, en las líneas 9, 10 y 11.

• Cuando se pulsa la tecla 6, la raqueta baja una línea, y sube cuando se pulsa la tecla 7. La función INKEY$ realiza esta operación.

Cuando la raqueta ocupa las líneas 19, 20 y 21, no puede bajar más, ya que se saldría de los límites de la pantalla. Tampoco podría subir si ocupara las líneas 0, 1 y 2. Para satisfacer estas dos condiciones, las líneas 260 y 270 contienen dos comparaciones separadas por la función lógica AND. Estas dos condiciones deben satisfacerse para que se ejecute THEN... Si t = 19 o t = 0, la función INKEY$ no funciona.

La línea 250 es necesaria para interrumpir el movimiento provocado por una de las teclas 6 ó 7.

Juego de squash

Para reunir las dos partes del programa de arriba, habrá que efectuar algunos arreglos.
He aquí el programa arreglado.

```
  10 LET x=INT (RND*20)+5 : LET y=
INT (RND*12)+5
  20 LET dx=1 : LET dy=1
  25 LET t=9 : LET dt=0
  30 PRINT AT y,x;" "
  40 LET x=x+dx : LET y=y+dy
  45 IF SCREEN$ (y,x)=CHR$ 124 THEN
LET dx=1
  50 PRINT AT y,x;"O"
  60 IF x>30 THEN LET dx=-1
  70 IF x<1 THEN GO TO 290
  80 IF y>20 THEN LET dy=-1
  90 IF y<1 THEN LET dy=1
 220 PRINT AT t,1;" ";AT t+1,1;" ";
AT t+2,1;" "
 230 LET t=t+dt
 240 PRINT AT t,1;CHR$ 124;AT t+1,1;CHR$ 124;
AT t+2,1;CHR$ 124
 250 LET dt=0
 260 IF INKEY$="6" AND t<19 THEN
LET dt=1
 270 IF INKEY$="7" AND t>0 THEN
LET dt=-1
 280 GO TO 30
 290 PRINT "Para una nueva pelota, pulsar
sobre RUN"
```

- La línea 10 se cambia para hacer aparecer la pelota, al azar, en la pantalla. (Véase la función RND en el anexo 2.)
- La línea 210 ha sido trasladada a la línea 25, ya que es una inicialización, lo mismo que las dos líneas precedentes.
- En la línea 45, la función SCREEN$ se utiliza para saber si la pelota ha tocado la raqueta, representada por caracteres 124, en cuyo caso la pelota rebota.
- A causa de esto, la línea 70 ha sido modificada. Si la pelota se dirige hacia la izquierda, y si llega a la columna 0 sin haber encontrado la raqueta, el juego se parará con un mensaje en la línea 290. Cambie este mensaje a su conveniencia.
- Evidentemente, se suprime la línea 100, y la línea 280 bifurca a la línea 30.
- En lugar de la línea 290, puede usted incrementar un contador de pelotas perdidas, después de haber inicializado este contador al inicio.

ALTA DEFINICIÓN

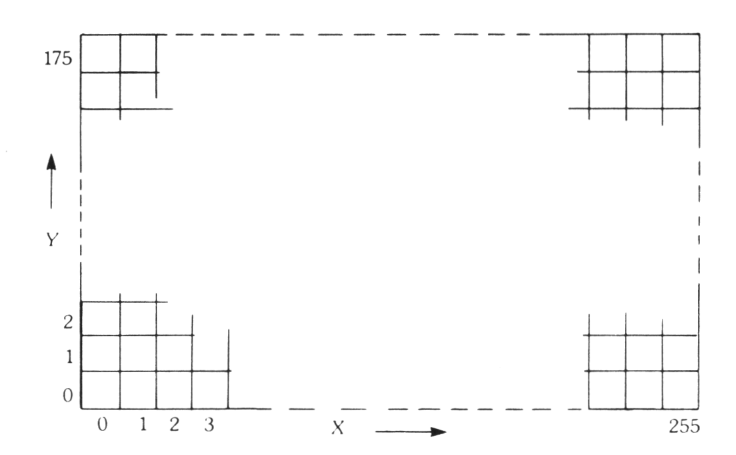
PLOT – DRAW – CIRCLE – función POINT

Hemos visto la división de la pantalla en posiciones PRINT, en 24 líneas de 32 caracteres. Una posición PRINT está ocupada por un carácter, de código ASCII, gráfico o UDG. Este carácter se encuentra almacenado en una matriz de 8 octetos de 8 puntos, denominados igualmente pixels. Así pues, **la pantalla se divide en 192 líneas de 256 columnas de pixels. Por lo que contiene 192 × 256, es decir 49152 puntos, pixels o posiciones PLOT.**

Al igual que las posiciones PRINT, las posiciones PLOT no tienen acceso a las dos líneas inferiores de la pantalla. Por consiguiente, disponemos de 176 líneas de pixels, numeradas de 0 a 175, y de 256 columnas, numeradas de 0 a 255.

La visualización y el borrado de los pixels se realiza mediante las instrucciones PLOT, DRAW y CIRCLE y la función POINT. Las coordenadas de un pixel son cartesianas.

Así pues, la pantalla se divide de la manera siguiente:



Las coordenadas de los 4 ángulos son:

(0,175)	(255,175)
(0,0)	(255,0)

En la inicialización, la posición PLOT se establece a (0,0). Las instrucciones CLEAR, CLS y RUN restablecen igualmente la posición PLOT a (0,0). Para completar, ni que decir tiene que NEW también lo hace.

La instrucción PLOT desplaza la posición PLOT y visualiza un punto en esta posición.

```
10 FOR n=1 TO 200
20 LET x=INT (RND*256) : LET y=INT (RND*176)
30 PLOT x,y
40 NEXT n
```

visualiza, al azar, 200 puntos o pixels.

La instrucción DRAW x1,y1 traza una línea desde la posición PLOT actual (x,y) hasta el punto de las coordenadas (x + x1,y + y1), que se convierte en la nueva posición PLOT actual.

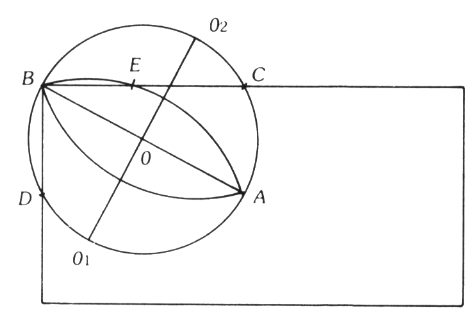
```
10 FOR n=1 TO 50
20 PLOT 128,88
30 DRAW INT (RND*256)-128,INT (RND*176)-88
40 NEXT n
```

Este programa dibuja 50 líneas, que parten del centro de las coordenadas (128,88) y desembocan en puntos de coordenadas aleatorias. Los argumentos de DRAW son tales, que los números generados serán positivos o negativos.
La abscisa va de –128 a 127 y la ordenada de –88 a 87.

Añadiendo un tercer argumento a la instrucción DRAW, expresado en radianes, el ordenador dibuja entonces un arco de círculo, que parte y desemboca en los mismos puntos. El arco de círculo, al igual que su ángulo del centro, está expresado en radianes por el 3.er argumento.

Para utilizar el pequeño programa precedente, deberemos tomar ciertas precauciones. Supongamos que trazamos una línea, del centro hacia el ángulo superior izquierdo, es decir, desde las coordenadas (128,88) hasta las coordenadas (0,175) con los mandatos:


PLOT 128,88: DRAW –128,87



Si queremos trazar un semicírculo con los mandatos:

```
PLOT 128,88 : DRAW -128,87,PI
```

el ordenador se para con un mensaje de error.

Se visualiza la parte de arco de círculo AC; más allá, la ordenada está fuera de límites. El centro 0 está en medio de AB.

Si tomamos un ángulo de arco más pequeño, el centro virtual se desplaza sobre la mediatriz de AB; hacia abajo, para los ángulos positivos, y hacia arriba, para los negativos.

```
PLOT 128,88 : DRAW -128,87,-PI
```

da un arco AD de igual centro 0.

```
PLOT 128,88 : DRAW -128,87,PI/2
```

dibuja un arco AE de centro 01

```
PLOT 128,88 : DRAW -128,87,-PI/2
```

dibuja un arco AB de centro 02.

He aquí, por último, un arco de círculo completo.

Sin entrar en consideraciones sobre el cálculo geométrico, lo que se sale del objetivo de esta obra, transformemos la línea 30 del último programa, para evitar salir de los límites de la pantalla:

```
30 DRAW INT (RND*256)-128,INT (RND*172)-86,PI/2
```

y el programa dibuja 50 arcos de círculo, al azar, de ángulo del centro de 90 grados.

Usted puede añadir artículos de color en la instrucción DRAW (véase anexo 1).

```
  30 DRAW INK (RND*6);INT (RND*256)-128,INT
(RND*172)-86,PI/2
```

La instrucción CIRCLE x,y,z dibuja un círculo de centro (x,y) y de radio z. Los tres parámetros se expresan en pixels.

```
  10 FOR n=1 TO 20
  20 CIRCLE 20+INT (RND*216),20+INT
(RND*136),20
  30 NEXT n
```

Este programa dibuja 20 pequeños círculos de 20 pixels de radio. Las coordenadas del centro se calculan de manera que queden dentro de los límites de la pantalla.

La función POINT (x,y) da 1 si el pixel (x,y) está visualizado y 0 si no lo está.

```
10 DRAW 255,175
20 LET x=0 : LET y=100
30 LET x=x+1
40 IF POINT (x,y) THEN PRINT x : STOP
50 PLOT x,y
60 GO TO 30
```

Este programa dibuja una línea diagonal y después una línea horizontal que comienza en (0,100).

Cuando esta línea encuentra la diagonal, el programa se para y se visualiza la abscisa.

Esto muestra la manera de detectar la coincidencia de 2 pixels y, luego, de intervenir en el sentido deseado.

SONIDOS

BEEP

La instrucción BEEP x,y produce un sonido de x segundos y de y semitonos, por encima del do central para y positivo, y por debajo, para y negativo.

```
  10 DATA x,y+4,2*x,y+4,x,y+2,2*x,y+2,
x,y+0,2*x,y+0
  11 DATA 2*x,y+2,x,y+4,2*x,y+5,x,y+4,
x,y+2,x,y+0,x,y+2,3*x,y+0
  20 LET x=.5
  30 LET y=0
  40 FOR n=1 TO 14
  50 READ a,b
  60 BEEP a,b
  70 NEXT n
```

Este programa toca una breve melodía. Las variables a y b se utilizan como argumentos de la instrucción BEEP. READ extrae estas variables de las listas de datos, situadas en declaraciones DATA. Estos datos están parametrados, lo que autoriza varias intervenciones.

Si usted desea que la melodía suene más rápida, haga:

```
20 LET x=.3
```

Si usted desea tocar la melodía en mi bemol, es decir 3 semitonos más alta, haga:

```
30 LET y=3
```

Para que suene una octava más alta, haga:

```
30 LET y=12
```

Si usted desea repetir esta melodía 4 veces seguidas, y cada vez una octava más alta, haga:

```
30 FOR y=-12 TO 24 STEP 12
80 RESTORE : NEXT y
```

Si usted desea repeticiones con tiempos diferentes:

```
20 FOR x=.5 TO .1 STEP -1
30 LET y=0
80 RESTORE : NEXT x
```

Para tocar las 12 notas de la escala cromática, sobre una extensión de 2 octavas, introduzca este programa:

```
10 FOR n=0 TO 23
20 BEEP .3,n
30 NEXT n
40 FOR n=22 TO 0 STEP -1
50 BEEP .3,n
60 NEXT n
70 BEEP 2,0
```

Para realizar arpegios, haga el programa:

```
10 DATA x,y,x,y+4,x,y+7
20 LET x=.2
30 FOR y=-12 TO 24 STEP 12
40 FOR n=1 TO 3
50 READ a,b: BEEP a,b
60 NEXT n
70 RESTORE : NEXT y
80 BEEP 1,36
```

Para cambiar la velocidad, usted asigna otro valor a x. Para cambiar el tono, usted aumenta y. Para 5 semitonos más alto, los límites del bucle y serán –7 y 29, y no olvide que el 36 de la línea 80 se convierte en 41.

Para producir un ruido de sirena:

```
10 FOR n=0 TO 15*PI STEP PI/30
20 BEEP.01,12+12*SIN n
30 NEXT n
```

Para producir un acompañamiento de vals:

```
10 LET x=.3
20 LET y=0
30 BEEP x,y : BEEP x/2,y+4 : BEEP x/2,69 : BEEP x,y+4
40 GO TO 30
```

Si desea usted un vals lento, haga:

```
10 LET x=.5
```

Si acompaña con un instrumento, póngase de acuerdo en el tono cambiando el valor de y.

ALGUNOS DIBUJOS

Creciente de luna

Dibujemos una media luna con la ayuda de 2 arcos:

```
10 LET x=230 : LET y=130
20 PLOT x,y
30 DRAW 0,40,PI/2
40 DRAW 0,-40,-PI
```

Para ennegrecer un creciente de luna, dibujemos los dos arcos de círculo con la ayuda de la ecuación cartesiana del círculo:

```
x2 + y2 = R2
```

de donde se calcula x

```
10 LET a=230 : LET b=150
20 FOR y=-20 TO 20
30 LET x2=SQR (400-y*y)
40 LET x1=SQR (784-y*y)
50 PLOT a+x1-18,b+y
60 PLOT a+x2,b+y
70 NEXT y
```

El primer arco tiene su centro en (a – 18,b) y su radio es de 28 (el cuadrado es 784).

El segundo arco tiene su centro en (a,b) y su radio es de 20 (cuadrado: 400).

Dibujemos pequeñas líneas desde el primer arco hasta el segundo. Partimos de la posición PLOT de la línea 50 para llegar a la posición de la línea 60. Los parámetros de la instrucción DRAW serán pues:

(a + x2) – (a + x1 –18) y (b + y) – (b +y)

Sustituyamos la línea 60 por

```
60 DRAW x2-x1+18,0
```

Para colorear el creciente de luna en amarillo, cambie las líneas 5 y 60:

```
50 PLOT INK 6;a-x1-18,b+y
60 DRAW INK 6;x2-x1+18,0
```

Puesta de sol

Añadamos una puesta de sol de color rojo, siguiendo el mismo principio:

```
100 FOR y=0 TO 20
110 LET x1=SQR (400-y*y)
120 PLOT INK 2;128-x1,y
130 DRAW INK 2;2*x1,0
140 NEXT y
```

Aquí, solamente tenemos un arco de círculo de centro (128,0) y de radio 20. Para cada valor de y, se dibujan líneas de 128 – x1 hasta 128 + x1.

Polígonos irregulares

El tercer parámetro de DRAW introduce polígonos irregulares, cuando este 3.er parámetro es suficientemente grande.

```
10 INPUT a
20 PLOT 128,10
30 DRAW OVER 1;0,155,PI*a
```

Cada vez que usted pulsa RUN, la línea 10 le solicita una introducción, asignada a la variable a. Ésta determina el 3.er parámetro de DRAW.

Intente las introducciones siguientes:

81, 105, 135, 147, 189, 243, 245, 343, 729

No utilice múltiplos de 2, el ordenador no los quiere, o traza una sola línea. Nosotros hemos utilizado los productos de tres números 3, 5 y 7. La línea 30 contiene OVER 1, para que el ordenador borre las líneas que ha visualizado diversas veces. Suprimiendo OVER 1, el dibujo se vuelve más denso.

Rectángulos

Dibujemos rectángulos; después, borrémoslos. Repitamos el proceso aumentando las dimensiones.

```
 10 FOR n=0 TO 150 STEP 10
 20 PLOT 78-n/2,98+n/2
 30 FOR m=0 TO 1
 40 OVER m
 50 DRAW 100+n,0 : DRAW 0,-(20+n)
: DRAW -(100+n),0 : DRAW 0,20+n
 60 IF n=150 THEN STOP
 70 NEXT m
 80 NEXT n
```

- El bucle controlado por m visualiza y luego borra un rectángulo.
- El bucle controlado por n aumenta las dimensiones del rectángulo.
- La línea 60 para terminar con la visualización de un rectángulo.

Círculos crecientes

```
10 PLOT 0,88
20 FOR n=0 TO 150 STEP 10
30 DRAW 20+n,0,PI : DRAW -(20+n),0,PI
30 NEXT n
```

Este programa dibuja 16 círculos, teniendo todos un punto común de coordenada (0,88). He aquí, otra manera de realizarlo con **la instrucción CIRCLE.**

```
10 FOR n=10 TO 85 STEP 5
20 CIRCLE n,88,n
30 NEXT n
```

Añadamos círculos a la derecha de la pantalla:

```
40 FOR n=245 TO 170 STEP -5
50 CIRCLE n,88,255-n
60 NEXT n
```

Añadamos círculos por arriba y por debajo de la pantalla:

```
25 CIRCLE 128,n,n
55 CIRCLE 128,n-80,255-n
```

Para que el efecto sea más sorprendente, añada la línea:

```
5 INK 7 : PAPER 0 : BORDER 0 : CLS
```

Elipse

Dibujemos una elipse utilizando las ecuaciones paramétricas.

```
10 FOR t=0 TO 2*PI STEP PI/50
20 LET x=100*COS t : LET y=50*SIN t
30 PLOT 128+x,88+y
40 NEXT t
```

Para llenar esta elipse de líneas, cambie la línea 30.

```
30 PLOT 128,88 : DRAW x,y
```

Todas las líneas parten del centro. Para modificar el comienzo, cambiemos la posición PLOT. Dejémosla, en primer lugar, en el interior de la elipse.

```
30 PLOT 28,88 : DRAW x+100,y
```

las interferencias también son muy bonitas.
Intente:

```
30 PLOT 128,38 : DRAW x,y+50
```

y he aquí una bonita copa.
Tomemos una posición PLOT fuera de la elipse.

```
30 PLOT 128,170 : DRAW x,y-82
```

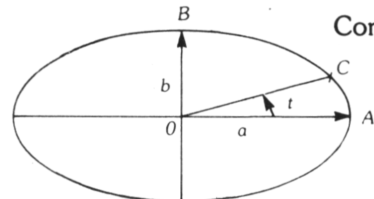
y he aquí un sombrero que las chinas se ponen para trabajar en los arrozales.

Para obtener un círculo, los ejes de la elipse deben ser iguales. Si en la línea 20, usted hace LET x = 50 * COS t, obtendrá un círculo.

Dibujemos un círculo y 4 elipses con el programa siguiente. Estúdielos, a fin de dominar todos los parámetros.

```
  5 INK 7 : PAPER 0 : BORDER 0 : CLS
 10 FOR t=0 TO 2*PI STEP PI/50
 20 LET x=40*COS t : LET y=40*SIN t
 30 PLOT 128,88 : DRAW x,y
 40 LET y=2*y
 50 PLOT 0,88 : DRAW x+40,y
 60 PLOT 255,88 : DRAW x-40,y
 70 LET x=2*x : LET y=y/4
 80 PLOT 128,175 : DRAW x,y-20
 90 PLOT 128,0 : DRAW x,y+20
100 NEXT t
110 PAUSE 0
```

Después de un examen, pulse una tecla para hacer aparecer el informe.



Como recordatorio, las ecuaciones paramétricas de la elipse son:

x = a.cos t
y = b.sin t

a es el eje semigrande OA
b es el eje semipequeño OB
t es el ángulo COA.

ROTACIONES Y DESPLAZAMIENTOS

En una pantalla ligeramente reducida a 240 × 160 pixels, hagamos girar una recta.

Al comienzo, se traza una recta horizontal, la cual irá inclinándose poco a poco, hasta convertirse en una vertical.

El punto de salida de cada recta avanza 10 pixels en cada circuito y el punto de llegada desciende 2/3 de 10 pixels para cada recta. El 1.er punto se desplaza sobre una horizontal y el 2.º sobre una vertical.

```
10 FOR d=0 TO 240 STEP 10
20 LET dy=d*2/3
30 PLOT 8+d,168
40 DRAW 240-d,0-dy
60 NEXT d
```

Aprovechando el punto de llegada, añadamos tres líneas para simular un rectángulo que se deforma.
Añadamos primero:

```
45 DRAW 0-d,-160+dy
```

observe, a continuación, añadamos:

```
50 DRAW -240+d,0+dy
55 DRAW 0+d,160-dy
```

Rotación de un cuadrado

```
10 LET t=128 : LET u=88 : LET a=80 : LET b=80
20 LET m=0
30 FOR n=0 TO 6.29 STEP PI/2
40 LET x=t-a*COS (n+m) : LET y=u-b*SIN (n+m)
50 PLOT x,y
60 IF n=0 THEN GOTO 80
70 DRAW x1-x,y1-y
80 LET x1=x : LET y1=y
90 NEXT n
```

Nosotros utilizamos las ecuaciones paramétricas del círculo:

$$\mathbf{x = r.cos\ n}$$
$$\mathbf{y = r.sin\ n}$$

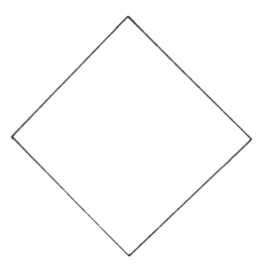
para determinar las coordenadas (x,y) de cada vértice sucesivo del cuadrado.

Se asignan a las coordenadas (x1,y1) los mismos valores, y la primera vez, para n = 0, se salta la instrucción DRAW. A continuación, esta instrucción traza una recta entre las coordenadas del punto anterior (x1,y1) y las coordenadas del nuevo punto (x,y). Las coordenadas (t,u) son las del centro de la pantalla.

- La línea 20 asigna el valor a m. Modificando m se producirán las rotaciones.

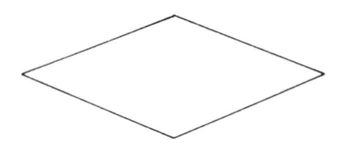
- En la línea 30, el límite superior de la variable de circuito es 6.29 para que el circuito se efectúe 5 veces. Este límite es ligeramente superior a 2 PI. Si hubiéramos puesto 2 * PI como límite, el circuito se habría efectuado cuatro veces y solamente habríamos obtenido tres lados del cuadrado.

Obtenemos un cuadrado, inscrito en un círculo virtual.



Como las ecuaciones paramétricas de la elipse difieren poco de las del círculo, tenemos la posibilidad de inscribir un polígono en una elipse virtual.

En la línea 10, cambie el valor de a, haga a = 120 y obtendrá un rombo:



Ahora hagamos variar todo lo que es susceptible de ser modificado.

Si se hace variar a, se obtiene una sucesión de rombos que se estiran horizontalmente. Si se hace variar b, los rombos se estiran verticalmente.

Si se desplazan las coordenadas (t,u) del centro, se desplaza el cuadrado o cuadrilátero, en el lugar de la pantalla que se desee.

Si se modifica el ángulo m, la figura efectúa rotaciones.

Si se introduce una nueva variable, se visualizan y luego se borran sucesivamente las figuras.

Veamos esto.

Introduzca las líneas siguientes:

```
 20 FOR m=0 TO PI STEP PI/8
100 NEXT m
```

Se dibujan 8 cuadrados, cada uno en un lugar distinto. Giran siguiendo un ángulo de rotación determinado por m. Algunos se superponen.

Borremos un cuadrado antes de visualizar el siguiente. Añada estas líneas:

```
25 FOR p=0 TO 1
27 OVER p
95 NEXT p
```

Se visualizan los 8 cuadrados, y después se borran.

Rombos que se deforman

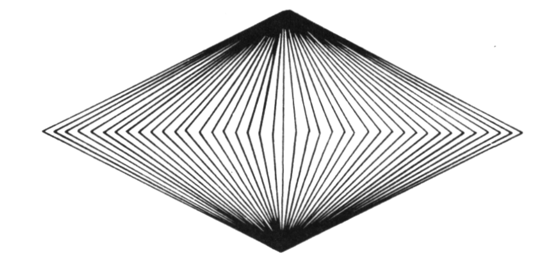
Suprimamos las líneas 20, 25, 27, 95 y 100; después, efectúe el mandato:

```
OVER 0
```

A continuación, añadamos las líneas:

```
 15 FOR a=0 TO 120 STEP 10
 20 LET m=0
100 NEXT a
```

Se obtiene un farolillo de verbena.



Añada las líneas:

```
20 FOR m=-PI/8 TO PI/8 STEP PI/4
95 NEXT m
```

y obtendrá dos farolillos ajustados de gran efecto.

Suprimamos las líneas 20 y 95, y añadamos:

```
 15 FOR b=0 TO 80 STEP 10
 20 LET m=0
100 NEXT b
```

Obtendremos otra figura.

Cuadrado que se desplaza

Suprima las líneas 15 y 100 y añada:

```
 25 FOR p=40 TO -40 STEP -10
 27 LET t=128-p
100 NEXT p
```

Obtendrá 9 cuadrados en punta.

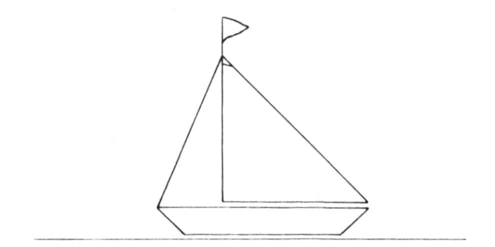
Barco que navega a merced del viento

```
  10 DRAW 255,0
  20 FOR n=0 TO 50 STEP 5
  30 LET a=0
  40 PLOT 128-n,11
  50 DRAW 60,0 : DRAW -5,-10 : DRAW -40,0
: DRAW -15,10
  60 DRAW 20,60 : DRAW 0,7 : DRAW 8,-4
: DRAW -8,-4 : DRAW 40,-58 : DRAW -40,0
: DRAW 0,58
  70 IF a=0 AND n<50 THEN LET a=1
: PAUSE 30 : GO TO 35
  80 NEXT n
```

• Las líneas 50 y 60 dibujan el barquito. El punto de partida del dibujo es la posición PLOT de la línea 40. Esta posición PLOT cambia en cada dibujo, gracias al circuito de las líneas 20 y 80.

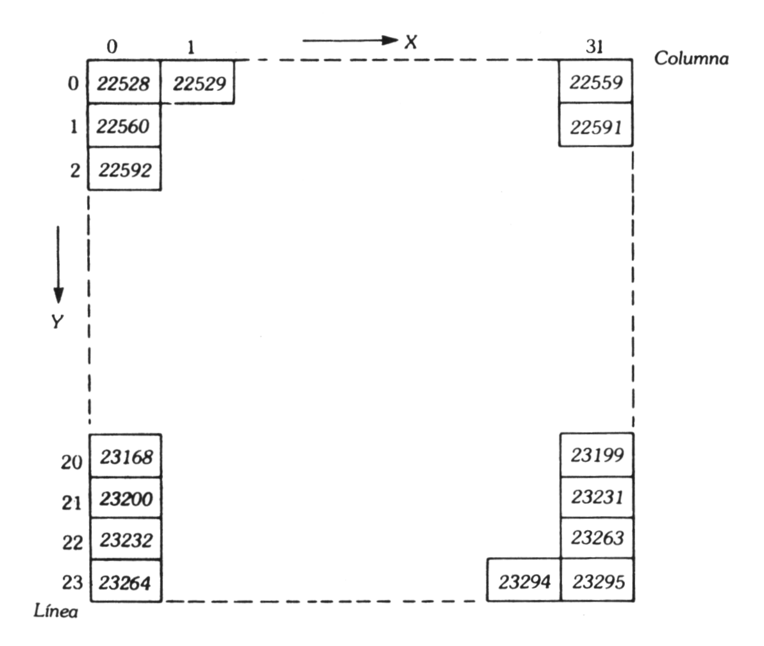
• Para visualizar y después borrar el barco antes de la visualización de la posición siguiente, se utiliza la instrucción OVER, pero sin ser el objeto de un circuito. Ya que dejamos la visualización del barco, antes de borrarlo, durante el tiempo de la instrucción PAUSE 30. Para esto, el parámetro a de la instrucción OVER a, se pone a 0 para dibujar el barco; después, se pone a 1 y después de la pausa se borra por bifurcación a la línea 35.

• Existe una 2.ª comprobación, en la línea 70, n < 50, para que el último barco visualizado no se borre.



ILUMINACIÓN

En el capítulo 4, Atributos, hemos visto que a cada posición PRINT le corresponde un juego de atributos posibles. Estos atributos afectan a toda la pantalla, cuando sus instrucciones se utilizan como declaraciones, o a una parte de la pantalla, cuando la instrucción apropiada de atributos está incluida en una instrucción tal como PRINT, INPUT, PLOT, DRAW, etc.
Por consiguiente, los atributos tienen la misma definición que los caracteres visualizables. Sin embargo, hay una pequeña mejora: puede asignarse un color tinta y un color papel en cada posición PRINT.
Los atributos posibles son el color tinta, el color papel, el brillo y el parpadeo (véase anexo 2). **Los atributos OVER e INVERSE no afectan a las células de memoria reservadas a los atributos.**
He aquí, la disposición de las células de memoria:



Para establecer la correspondencia con las posiciones PRINT, hemos puesto igualmente las posiciones de 32 columnas y de 24 líneas. Las células de la memoria que contienen los atributos se suceden desde 22528 hasta 23295. He aquí un programa que ilustra lo anterior:

```
 10 FOR n=0 TO 31
 20 PRINT INK 2; PAPER 5; CHR$ 131;
 30 NEXT n
100 FOR n=22560 TO 22591
110 PRINT CHR$ 131; : POKE n,42
120 NEXT n
```

• El circuito 10 a 30 llena la línea 0 de caracteres gráficos de código 131, poniendo tinta roja y papel cyan.

• El circuito 100 a 120 hace lo mismo con la línea 1. Los atributos son 2 + 8 veces 5, es decir 42 (2.° argumento de POKE) (Véase la palabra ATTR en el anexo 2).

Para llenar la pantalla de cuadrados cuyo color papel se elige al azar:

```
10 FOR n= 22528 TO 23295
20 POKE n,(RND*7+1)*8
30 NEXT n
40 PAUSE 0
```

• La línea 40 le permite ver toda la pantalla llena de pequeños cuadrados de color. Cuando usted pulse una tecla, aparecerá el mensaje de informe y las dos últimas líneas volverán al color de los márgenes.

• Esto no es posible con la instrucción PRINT, ya que esta instrucción está subordinada a las comprobaciones de validez que el ordenador efectúa en sus argumentos. Llenando directamente las células de la memoria, con la ayuda de la instrucción POKE, las dos últimas líneas serán accesibles.

Para visualizar franjas de color variable y de longitud variable.

```
10 RANDOMIZE
20 LET d=22528+INT (RND*768)
30 LET L=RND*15 : LET C=(1+RND*7)*8
40 IF d+L>23295 THEN GO TO 20
50 FOR n=d TO d+L
60 POKE n,C
70 NEXT n
80 GO TO 20
```

• La línea 40 evita introducir datos en las células de la memoria más altas de 23295.

Para visualizar franjas de color verticales, en lugar de franjas horizontales, introduzca las 2 líneas:

```
40 IF d+32*L>23295 THEN GO TO 20
50 FOR n=d TO d+32*L STEP 32
```

y franjas verticales de colores y de longitudes variables llenarán la pantalla.

Para visualizar al azar franjas verticales u horizontales, introduzca las 3 líneas:

```
35 LET x=1+31*INT (RND*2)
40 IF d+x*L>23295 THEN GO TO 20
50 FOR n=d TO d+x*L STEP x
```

Para visualizar rectángulos de dimensiones y colores variables, deberemos utilizar PLOT y DRAW, lo que nos impedirá acceder a las 2 últimas líneas de la pantalla.

```
10 RANDOMIZE
20 LET L=INT (RND*80) : LET H=INT (RND*80)
30 LET X=INT (RND*256) : LET Y=INT (RND*176)
40 IF X+L>255 OR Y+H 175 THEN GO TO 20
50 LET C=INT (RND*8)
60 FOR n=Y TO Y+H
70 PLOT X,n : DRAW INK C;L,0
80 NEXT n
90 GO TO 20
```

Evidentemente, como el color no tiene la misma definición que el pixel, algunos rectángulos son incompletos.

Tapiz persa

```
 5 BORDER 0 :OVER 1
10 LET a=INT (RND*176) : LET b=INT (RND*7)
20 FOR n=a TO 175 STEP 4
30 PLOT n,n
40 LET x=255-2*n : LET y=175-2*n
50 DRAW INK b;0,y : DRAW INK b;x,0
: DRAW INK b;0,-y : DRAW INK b;-x,0
60 NEXT n
70 GO TO 10
```

Observe los bonitos tapices constantemente renovados.

Pare el programa en un momento preciso, si desea examinar un tapiz determinado.

Saturno y sus anillos

```
  5 BORDER 1 : PAPER 1 : INK 5 : CLS
 10 FOR Y=-40 TO 40
 20 LET X=SQR (1600-Y*Y)
 30 PLOT 128-X,78+Y
 40 DRAW 2*X,0
 50 NEXT Y
 60 FOR A=60 TO 120 STEP 20
 70 FOR N=0 TO PI STEP PI/25
 80 LET X=A*COS N
 90 LET Y=A/4*SIN N
100 PLOT OVER 0;128+X,78+Y
 PLOT OVER 1;128-X,78-Y
110 NEXT N
120 NEXT A
130 PAUSE 0
```

Las explicaciones dadas en el capítulo anterior hacen que este programa sea comprensible.

- El circuito 10 a 50 dibuja un círculo completo.
- El circuito 70 a 110 dibuja una elipse por puntos.
- El circuito 60 y 120 dibuja 4 elipses.

Cada elipse está dibujada en dos veces, para que los puntos de la cara oculta no aparezcan, pero sí los de la parte delantera, gracias a la instrucción OVER.

MINILOGO

Una de las particularidades del lenguaje evolucionado LOGO es permitir, con la ayuda de algunas **"primitivas"**, dibujar en la pantalla todos los motivos deseados. Esta característica es uno de sus principales atractivos.

El programa que viene a continuación, aunque solamente contiene 6 primitivas, le permitirá familiarizarse con este lenguaje, dibujando motivos, polígonos, círculos, etc.

Antes de explicar el programa, su funcionamiento y su modalidad de utilización, facilitamos su listing.

Minilogo

```
  10 INPUT "Para el minilogo, introduzca:
mini",a$
  15 IF a$="mini" THEN GO TO 5000
  20 PRINT AT 0,0;"        " : REM  8 espacios
4999 STOP
5000 REM   INICIALIZACION
5001 REM
5010 CLS : PRINT "minilogo"
5015 LET x=128 : LET y=88
5020 LET dx=0 : LET dy=1 : LET
an=PI/2 : LET indic=0
5025 PLOT x,y
5030 REM      INTERPRETE
5031 REM
5040 OVER 1 : GO SUB 5900
5050 INPUT "Introduzca su mandato",a$
5055 GO SUB 5900 : OVER 0
5060 LET der=LEN a$
5065 IF CODIGO a$<65 OR CODIGO a$(der)
=32 THEN GO TO 5040
5070 LET com=CODIGO a$+CODIGO a$(2)
5075 FOR n=1 TO der
5080 IF CODIGO a$(n)=32 THEN LET
para=VAL a$(n TO der)
5090 NEXT n
5095 IF com<194 THEN LET com=
```

```
com+64
5100 GO TO 5000+com
5194 GO TO 5040
5195 GO TO 20
5198 GO TO 5040
5199 GO TO 5450
5203 GO TO 5250
5209 GO TO 5400
5215 GO TO 5300
5227 GO TO 5350
5250 REM      BORRA
5251 REM
5255 IF para>255 THEN GO TO 5000
5260 OVER 1 : LET para=-para
5300 REM     AVANZA
5301 REM
5305 DRAW para*dx,para*dy : OVER 0
5310 IF indic=1 THEN GO TO 5485
5315 GO TO 5030
5350 REM    GIRA
5351 REM
5355 LET an=an+PI/180*para
5360 IF an>=PI*2 THEN LET
an=an-PI*2
5365 LET dx=COS an : LET dy=SIN an
5370 IF indic=1 THEN GO TO 5490
5375 GO TO 5030
5400 REM        LEVANTA
5401 REM
5405 LET x=INT para
5410 LET y=(para-x)*1000
5415 IF x>255 OR y>175 THEN GO TO 5030
5420 GO TO 5020
5450 REM       HACER
5451 REM
5455 IF indic=1 THEN GO TO 5465
5460 LET vez=para : LET indic=1
: GO TO 5030
```

```
5465 LET depla=INT para
5470 LET rot=(para-depla)*1000
5475 FOR n=1 TO veces
5480 LET para=depla : GO TO 5300
5485 LET para=rot : GO TO 5350
5490 NEXT n
5495 LET indic=0
5500 GO TO 5030
5900 REM   CURSOR DE POSICION
5901 REM
5910 DRAW 4*COS (an+PI/2),4*SIN (an+PI/2)
: DRAW 8*COS (an-PI/6),8*SIN (an-PI/6)
: DRAW 8*COS (an-PI*5/6),8*SIN (an-PI*5/6)
: DRAW 4*COS (an+PI/2),4*SIN (an+PI/2)
5920 RETURN
```

El programa

• La línea 10 le pide que introduzca 'mini', si usted desea el programa minologo. Introduciendo mini, la línea 15 efectúa un salto a la 5000. La primitiva 'Basic' vuelve a dar el control a su programa Basic, gracias a la línea 5195 que bifurca a la línea 20. De está forma, usted puede desarrollar, cuando lo desee, un programa Basic con acceso al minilogo. Para que el retorno del minilogo a su programa no se efectúe siempre en la línea 20, cambie la línea 5195 GO TO a.

• Prevea en su programa Basic un indicador al que usted asignará un valor de retorno, antes de la llamada del minilogo, que se efectuará por GO TO 5000.

• De 5000 a 5025 es la inicialización. Se borra la pantalla, se visualiza 'minilogo', que quedará visualizado mientras usted no vuelva otra vez al Basic; después se asignan valores iniciales a algunas variables.

• Las líneas 5030 a 5227 contienen el intérprete, al que usted vuelve después de cada mandato. Éste debe contener una primitiva, después un espacio, por lo menos, y luego un número. Este último es optativo en algunos casos.

• La línea 5040 visualiza el cursor, representado por un pequeño triángulo, cuya punta indica la dirección.

• La línea 5050 introduce su mandato.

• La línea 5055 borra el cursor.

• La línea 5070 asigna a la variable 'com', la suma de los códigos de las 2 primeras letras de la primitiva.

• El circuito 5075 a 5090 busca la l.ª ocurrencia del carácter 'espacio'. Cuando se ha encontrado este carácter, se asigna a la variable 'para' el valor del número presente en el mandato.

• La línea 5095 permite introducir la primitiva, ya sea en minúsculas o en mayúsculas.

• La línea 5100 bifurca a 5000 + com, que es la suma de los códigos de las 2 primeras letras, y de allí, a la rutina de la primitiva escogida.

• Las líneas 5250 a 5260 contienen la primitiva 'borra'. Cuando el número que sigue a 'borra' es mayor de 255, corresponde al mandato 'borra todo' (línea 5255).

• La línea 5260 invierte la variable 'para', después se entra en la rutina de la primitiva 'avanza'.

• La línea 5305 dibuja la línea solicitada.

• La línea 5310 es necesaria para la ejecución de la primitiva 'hacer'.

• La línea 5315 vuelve al intérprete, que espera el mandato siguiente.

• Las líneas 5350 a 5375 contienen la primitiva 'gira'.

• La línea 5355 cambia el ángulo 'an' en función del parámetro del mandato, con la conversión de los grados en radianes.

• La línea 5360 disminuye 'an' de 2PI, si el ángulo es mayor de 2PI. Esta línea no es necesaria, pero si usted gira todo el tiempo en el mismo sentido, 'an' tomará un valor cada vez mayor, cosa que, de todas formas, las funciones trigonométricas aceptan.

• La línea 5365 prepara dx y dy para la utilización en la línea 5305.

• La línea 5370 es necesaria para la ejecución de la primitiva 'hacer'.

- La línea 5375 bifurca al intérprete.
- Las líneas 5400 a 5420 contienen la primitiva 'levanta'.
- Las líneas 5405 y 5410 separan los dos parámetros, que se introducen al mismo tiempo –separados por un punto– como números en el mandato.
- La línea 5420 regresa a la parte 'inicialización' apropiada y, a continuación, al intérprete.
- Las líneas 5450 a 5500 contienen la rutina de la primitiva 'hacer'.
- La línea 5460 asigna el valor de 'para' a la variable 'vez', en la primera pasada. Después, 'indic' se pone a 1, y luego, regresa al intérprete.
- En la segunda pasada, la línea 5455 obliga a dar un salto a la línea 5465, en donde esta línea y la siguiente 5470 asignan las variables 'depla' y 'rot' como lo hemos hecho en 'levanta' para x e y.
- El circuito 5475 a 5490 efectúa 'avanza', después 'gira', tantas veces como se ha solicitado en la primera pasada. Por esta razón, las primitivas 'avanza' y 'gira' vuelven a este circuito cuando indic = 1.
- Las líneas 5495 y 5500 vuelven a poner indic a 0; después retornan al intérprete.
- La línea 5910 dibuja el pequeño cursor, representado por un triángulo orientado.

Las comprobaciones del programa

Se realizan un determinado número de comprobaciones para evitar la salida del programa 'minilogo'.

- La línea 5065 evita que usted introduzca su mandato con un carácter distinto de una letra. Si su primer carácter es un espacio (se modificaría la variable 'com'), esta línea le envía a 5040. O si el último carácter del mandato es un espacio (no se ejecutaría la función VAL de la línea 5080), la línea 5065 le enviaría a 5040.
- La línea 5194 le envía a 5040 si, por casualidad, la línea 5100 efectúa un salto entre 5100 y 5195. Si el salto imprevisto (debido a la decodificación de las 2 primeras letras de un mandato no previsto) le lleva entre 5195 y 5244 (5000 + dos veces el código de z), usted entra en una rutina. Para evitar que sea la rutina 'hacer', se ha colocado igualmente la línea 5198.
- La línea 5415 impide que se salga fuera de la pantalla.

A pesar de todas las comprobaciones, existen algunas posibilidades de interrupción con mensaje de error: si los parámetros de DRAW superan los límites de la pantalla; o si usted introduce 40,40 en lugar de 40.40 (véase más adelante: utilización de una coma en lugar de un punto). En todos los casos en que usted obtenga una interrupción, introduzca el mandato:

```
GO TO 5015
```

que no modifica en nada su dibujo y sitúa el cursor en el centro de la pantalla.

Funcionamiento

Las primitivas son 6.

– avanza n

donde n es un número entero, que establece el número de pixels para el desplazamiento del cursor. Al desplazarse, el cursor dibuja una línea en la dirección indicada con su punta.

– gira m

donde m es un número entero, comprendido entre –360 y 360, que expresa el número de grados que usted desea que gire el cursor. Los ángulos positivos desplazan el cursor en el sentido trigonométrico (al contrario del reloj), los ángulos negativos en el otro sentido.

– borra n

Esta primitiva puede introducirse sin su parámetro n. En este caso, borra la línea que usted acaba de trazar, sin cambiar el sentido del cursor.

Utilizado con su parámetro n, borra n pixels de la línea que acaba de ser trazada.

Por último, si n es mayor que 255, por ejemplo el mandato minilogo:

```
borra 300        ENTER
```

borra toda la pantalla y sustituye el cursor en el centro.

– levanta x.y

Esta primitiva borra el cursor y lo lleva a las coordenadas (x,y), sin trazar ninguna línea. Las coordenadas (x,y) serán la nueva posición PLOT donde se visualizará el cursor. x va de 0 a 255 e y de 0 a 175. Para y, es absolutamente necesario introducir 3 cifras. Si usted desea visualizar el cursor en las coordenadas (200,5), introduzca el mandato minilogo:

```
levanta 200.005
```

Existen ciertas limitaciones en x y en y, debidas al dibujo del cursor (8 pixels de ancho y 7 de alto).

Dado que puede desplazarse el cursor adonde se quiera sin trazar ninguna línea, no habrá necesidad de la primitiva 'baja'.

– basic

Esta primitiva bifurca al programa Basic

– hacer t seguido de

hacer n.m

Esta primitiva efectúa t veces los 2 mandatos

```
avanza n
gira m
```

Al igual que en 'levanta', el parámetro m debe escribirse con 3 cifras. Además, m debe ser positivo. Esto no es nada molesto, ya que si usted desea girar –90 grados es el mismo resultado que girar 270 grados.

Observación: Para todas estas primitivas, usted solamente podrá introducir las dos primeras letras de la primitiva, en lugar de la palabra completa.

Ejemplos

Dibujos

Con las primitivas 'avanza n' y 'gira m', usted dibujará los motivos que desee. Por ejemplo, **para dibujar una caja en perspectiva,** introduzca los mandatos minilogo:

```
avanza 30
gira   90
avanza 40
gira   90
avanza 30
gira   90
AVANZA 40
GIRA   45
AVANZA 50
GIRA   45
avan 30
gir 90
av 40
gi 45
AV 50
levanta 128.118
gira    -45
avanza  50
levanta 10.010
```

Polígonos estrella

Utilización de la primitiva 'hacer'.

```
borra 300
hacer 5
hacer 20.072
```

Dibuja un pentágono

Para dibujar uno más grande:

```
hacer 5
hacer 40.072
```

Para el pentágono, se divide 360 entre 5 para encontrar el ángulo de 72 grados.
Dibujemos **una estrella de 5 puntas:**

```
borra 400
hacer 5
hacer 60.144
```

Círculos

```
borra 300
hacer 36
hacer 10.010
```

para dibujar otro más pequeño

```
hacer 36
hacer 5.010
```

a continuación, otro a la derecha:

```
hacer 36
hacer 5.350
```

Anexo 1
RECAPITULACIÓN DE LAS INSTRUCCIONES BASIC

Los parámetros de las instrucciones tienen el significado siguiente:

a	representa una letra
v	representa una variable (el primer carácter siempre es una letra)
x,y,z	representan expresiones numéricas
m,n	representan expresiones numéricas que el ordenador redondea al entero más próximo
e	representa una expresión
f	representa una expresión de cadena
s	representa una secuencia de declaraciones, separadas por :
c	representa uno o varios artículos de color, separados por , o por ;
p	representa una secuencia de artículos PRINT

DATA, DEF e **INPUT** se utilizan solamente como declaración (en una línea de programa que lleve un número) y no como mandato directo, lo que permiten indiferentemente las otras instrucciones.

Los artículos de color c pueden ser **PAPER, INK, FLASH, BRIGHT, INVERSE, OVER.** Los artículos de color c presentes en una instrucción que los permita, solamente afectarán a la o en las posiciones PRINT referidas por esta instrucción.

Toda línea Basic, mandato o declaración, debe empezar por una instrucción.

BEEP x,y

produce un sonido de x segundos, en una frecuencia de y semitonos, por encima (para y positivo) y por debajo (para y negativo) del do central.

```
BEEP 1,0
```

emite un do central, es decir, el del centro del teclado del piano, durante un segundo.

BORDER m

pone el color m en los márgenes de la pantalla y color papel en la parte inferior de la pantalla (zona de edición). m está comprendido entre 0 y 7, y los colores correspondientes se imprimen por encima de estas cifras.
El ordenador redondea la fracción de m.

```
BORDER 2
```

pone el color rojo en los márgenes.

BRIGHT m

los caracteres visualizados más tarde se asignan de la manera siguiente:
m = 0 para brillo normal
m = 1 para brillo resaltado
m = 8 para brillo transparente

```
BRIGHT 1
```

pone los caracteres visualizados más tarde en brillo resaltado.

CAT

reservado a los periféricos tales como microdrive, etc.

CIRCLE x,y,z

dibuja un círculo de centro (x,y) y de radio z. Los tres parámetros se expresan en pixels.

```
CIRCLE 128,96,50
```

dibuja un círculo cuyas coordenadas del centro son (128,96) y el radio 50.

CIRCLE c;x,y,z

ídem pero con artículos de color c.

CLEAR

efectúa las operaciones siguientes:

- borra las variables
- ejecuta RESTORE
- ejecuta CLS
- sitúa la posición PLOT al inicio:
  en las coordenadas (0,0)
- borra la pila GOSUB

CLEAR m

efectúa las mismas operaciones CLEAR y, además, proporciona el valor m a la variable del sistema RAMTOP

```
CLEAR 32000
```

pone RAMTOP a 32000

CLOSE #

reservado a los periféricos: microdrive, etc.

CLS

borra el fichero de visualización y pone la posición PRINT a 0,0.

CONTINUE

vuelve a tomar el programa interrumpido por un STOP o por un BREAK. Está abreviado (CONT) sobre la tecla C.

COPY

envía las 22 primeras líneas visualizadas en la pantalla hacia la impresora, siempre, por supuesto, que la impresora esté conectada.

DATA e1, e2, e3...

Las expresiones e1, e2, e3, etc., numéricas o de cadena, forman una lista de datos. Solamente se utiliza en una declaración.

```
100 DATA 100,50,"buenos días"
```

esta declaración forma una lista de 3 datos: dos números y una cadena.

DEF FN a (a1, a2,...) = e

permite al usuario definir una función de nombre a por una numérica, y a$ por una función de cadena. Los argumentos son también numéricos a1, a2, etc., o de cadena a1$, a2$, etc., e es una expresión numérica o de cadena. Si no existen argumentos, el formato será

```
DEF FN a()=e
```

DEF se utiliza solamente como declaración.

```
150 DEF FN a(X,Y)=SQR (X*X+Y*Y)
```

calcula la hipotenusa cuando se facilitan los dos lados del ángulo recto.

```
PRINT FN a(3,4)
```

visualiza 5.

DIM a(m1,m2,...,mk)
DIM a$(m1,m2,...,mk)

Reserva los espacios de memoria necesarios para una matriz numérica de nombre a y de dimensión K, o por una matriz de cadena de nombre a$ y de dimensión K. La última dimensión puede considerarse como la longitud máxima de cada cadena. Inicializa cada valor numérico a 0 o cada cadena a " ".

```
DIM A(5,5,5)
```

es una matriz de 3 dimensiones de nombre A. Contiene 125 números inicializados a 0.

```
DIM A$(5,12)
```

es una cadena de 2 dimensiones de nombre a$. Contiene 5 cadenas de 12 caracteres como máximo inicializadas a " ".

DRAW x,y,z

traza un arco de círculo que parte de la última posición PLOT y llega al punto donde las coordenadas son las de PLOT, aumentadas en x,y, girando un ángulo z. x e y se expresan en pixels y z en radianes. Cuando se omite z, se reduce al formato DRAW x,y, el cual, entonces, traza una línea.

```
PLOT 40,40
DRAW 200,100
```

traza una línea desde el punto (40,40) al punto (240,140).

```
PLOT 90,90
DRAW 150,0,PI
```

traza un semicírculo hacia abajo, partiendo de (90,90) y llegando a (240,90).

DRAW c;x,y,z

ídem con los artículos de color c.

ERASE

reservado a los periféricos: microdrive, etc.

FLASH m

Los caracteres visualizados más tarde, se asignarán de la manera siguiente:
m = 0 inmóviles
m = 1 parpadeantes
m = 8 sin cambio

FOR a=x TO y STEP z

Asigna a la variable de control a (letra simple) el valor x, el límite y el paso z. Cuando no se utiliza STEP z, el paso será de 1. Esta instrucción provoca un circuito en el programa que va de la instrucción siguiente hasta encontrar la instrucción NEXT a. Este circuito se efectuará hasta que haya alcanzado o superado su límite.

```
FOR n=2 TO 11 STEP 2
```

ejecutará 5 circuitos.

FORMAT

reservado a los periféricos: microdrive, etc.

GO SUB m

Guarda el número de línea, que contiene GO SUB, en la pila GO SUB; después, efectúa GO TO m.

GO TO m

salta a la línea m, o a la siguiente, si la línea m no existiera.

```
GO TO 1000
```

efectúa un salto a la línea 1000.

IF x THEN s

Si x es verdadero, entonces se ejecuta s.
Si x es falso, se pasa a la línea siguiente.

```
IF x=1 THEN PRINT a : LET a=0
```

visualiza la variable a, después la pone a 0, si x es igual a 1.

INK m

Los caracteres visualizados a continuación, serán de color m. Además de los valores 0 a 7 (los colores correspondientes se imprimen por encima de estas teclas), m puede tomar los valores:
m = 8 transparente
m = 9 contrastado: que proporciona caracteres cuyo color, blanco o negro, contrastará automáticamente con el color papel utilizado.

INPUT p;v

espera una entrada desde el teclado. p es una secuencia de artículos PRINT. p puede omitirse. v es una variable numérica o de cadena a la que se asigna la entrada. Los artículos PRINT, si existieran, aparecerían en la parte inferior de la pantalla.

```
INPUT edad
```

en la ejecución, usted obtiene una <u>*L*</u> parpadeante, e introduce un número

```
INPUT a$
```

en la ejecución, usted obtiene una <u>*"L"*</u> que parpadea, e introduce una cadena.

INPUT "Introduzca su edad"; edad
introduzca 32, y la variable edad contendrá 32. En los artículos PRINT, podemos utilizar , ; ' , al igual que en la instrucción PRINT.

INPUT "¿Está" ' "contento?" ' a$
introduzca sí

INPUT LINE v$

Da una _L_ parpadeante, sin sus comillas, aunque la entrada es una cadena.

INVERSE m

Los caracteres visualizados luego, serán
m = 0 en video normal
m = 1 en video invertido

LET v=e

se asigna el valor e a la variable v. Para una variable de cadena, indexada o dividida, la asignación es procusteana (truncada de acuerdo con la longitud de la variable de cadena). Los únicos medios de asignar un valor a una variable son LET, INPUT y READ.

```
LET r=5
PRINT r                         visualiza 5
LET b$="bueno"
PRINT b$                        visualiza bueno
DIM a$(2,5)
LET a$(1)="tamaño"
PRINT a$(1)
```

LIST m

lista el programa partiendo de la línea m o de la siguiente, si no existe la línea m. Cuando se omite m, se lista todo el programa. El apuntador de línea se desplaza a m o a 0, si no existe m.

LLIST m

lo mismo que LIST, pero sobre impresora, en lugar de visualizar en la pantalla.

La instrucción LOAD, que significa cargar, transfiere informaciones desde el grabador hacia el ordenador. Estas informaciones han sido previamente salvadas con la instrucción SAVE, desde el ordenador hacia el grabador. Esta instrucción es tan importante, que detallamos todas sus posibilidades.

LOAD "f"

borra el programa y las variables presentes en el ordenador, y carga el programa de nombre f y las variables asociadas.

```
LOAD "juego"
```

carga el programa de nombre juego.

```
LOAD ""
```

carga el primer programa que se encuentra en la cassette.

LOAD "f" DATA a()

carga la matriz numérica a.

```
LOAD "mat" DATA x()
```

carga la matriz numérica x de nombre mat.

LOAD "f" DATA a$()

carga una matriz de cadena a$

```
LOAD "dic" DATA x$()
```

carga la matriz de caracteres x$ de nombre dic.

LOAD "f" CODE m,n

carga n octetos a partir de la dirección m.

```
LOAD "carac" CODE 32600,168
```

carga 168 octetos y los coloca en las direcciones 32600 a 32767, de nombre carac.

Para un Spectrum de 16 k, se trata aquí de los 21 caracteres gráficos definidos por el usuario.

LOAD "f" CODE m

carga los octetos partiendo de la dirección m.

```
LOAD "carac" CODE 32600
```

carga igualmente 168 octetos para un Spectrum de 16 k; pero carga 32 k de octetos de más, para un 48 k.

LOAD "f" CODE

carga todos los octetos hasta la dirección donde éstos han sido salvados.

LOAD "f" SCREEN$

carga el fichero de visualización y la pantalla se llena automáticamente durante la carga.

Equivalen a:

```
LOAD "f" CODE 16384,6912
```

LPRINT

como PRINT, pero con la impresora. Ésta ignora el número de línea de un LPRINT AT. Puede imprimir más de 22 líneas.

MERGE "f"

como LOAD "f", pero no borra el programa ni las variables que ya se encuentran en el ordenador, con la excepción de las líneas de programa que llevan los mismos números y las variables de igual nombre.

Este mandato sirve, pues, para añadir líneas a un programa existente o para mezclar 2 programas. No puede utilizarse MERGE para matrices ni para octetos.

```
MERGE "renum"
```

añade el programa de renumeración, de nombre renum, a su programa en curso de elaboración.

MOVE

reservado a los periféricos: microdrive, etc.

NEW

vuelve a arrancar el sistema Basic, borrando toda la memoria hasta e incluso el octeto cuya dirección se encuentra en RAMTOP. Las variables del sistema RAMTOP, UDG, P–RAMT, RASP y PIP se encuentran protegidas.

Si usted desea conservar octetos para utilizarlos en el programa siguiente, y si estos octetos están situados en la dirección 32000 y siguientes, haga CLEAR 31999, y después NEW. Por otro lado, para un arranque completo parecido al que se produce en la conexión, haga

```
RANDOMIZE USR 0
```

NEXT a

instrucción de cierre de un circuito

```
FOR a=x TO y STEP z
```

NEXT añade a la variable de control a el paso z. Si la variable de control no ha alcanzado el límite y, vuelve a iniciarse el circuito en la línea que sigue a la instrucción FOR.
En caso contrario, el programa continúa con la instrucción que sigue a NEXT a.

OPEN

reservado a los periféricos: microdrive, etc.

OUT m,n

saca el octeto n hacia la boca de salida m, en el microprocesador (carga BC con m, carga A

con n, después ejecuta OUT (C),A. Esto sigue una terminología propia al ensamblaje de los códigos máquina).

OVER m

controla la sobreimpresión de los caracteres visualizados a continuación, si lo están en posiciones PRINT ya ocupadas por otros caracteres.
m = 0 el nuevo carácter sustituye al antiguo
m = 1 el nuevo carácter sobreimprime al antiguo

```
PRINT "1234567890"
PRINT OVER 1; AT 0,0;"qwertyuiop"
```

vea la sobreimpresión.

PAPER m

PAPER m es, en el fondo, lo que INK m es al carácter visualizado. m puede tomar los valores 0 a 9 (véase en INK).

```
PAPER 5
```

convertirá el fondo a color cyan.

PAUSE m

para el desarrollo del programa durante m quincuagésimas de segundo.

```
PAUSE 100
```

para el programa durante 2 segundos

```
PAUSE 0
```

para el programa hasta que se pulse una tecla.

PLOT m,n

visualiza un pixel en las coordenadas (m,n). El ordenador toma los valores absolutos de m y de n, después de haberlos redondeado.

```
PLOT 2*64,-88.2
```

es válido. Visualiza un pixel central.
La posición PLOT se desplaza a las coordenadas del pixel.

PLOT c;m,n

ídem con artículos de color.

POKE m,n

escribe el valor n en el octeto de dirección m.

```
POKE 16384,255
```

visualiza una raya de 8 pixels en la parte superior izquierda de la pantalla. (16384 es el primer octeto del fichero de visualización.)

PRINT p

los artículos PRINT, así como los separadores, desplazan la posición PRINT. Un artículo PRINT puede ser una expresión numérica. Si el número es demasiado grande o demasiado pequeño, el ordenador se sitúa en notación exponencial.

```
PRINT 123
PRINT 123456789
```

Un artículo PRINT puede ser una variable numérica que ha sido asignada.

```
LET un=1
PRINT un
```

Un artículo PRINT puede ser una cadena de caracteres. La cadena debe ponerse entre comillas.

```
PRINT "Buenos días amigo"
```

Un artículo PRINT puede ser una variable de cadena que ha sido asignada.

```
LET a$="Spectrum"
PRINT a$
```

Cada cifra o letra de estos artículos desplaza la posición PRINT de una posición. Los separadores desplazan la posición PRINT de la manera siguiente:

; deja la posición PRINT sin cambiar
, desplaza la posición PRINT a la columna 16.

o, si es imposible, a la columna 0 de la línea siguiente.
' desplaza la posición PRINT a la línea siguiente
nada desplaza la posición PRINT a la línea siguiente
AT m,n; desplaza la posición PRINT a la línea m, columna n
TAB m; desplaza la posición PRINT a la columna m

```
PRINT 2;4
PRINT 2,4
PRINT 2'4
PRINT
PRINT 3
PRINT AT 11,16;8
PRINT TAB 25;"AH"
```

Las expresiones numéricas y de cadena no están alineadas por un espacio, ni a derecha ni a izquierda. Es necesario preverlo si desea usted airear el texto.

```
PRINT 20;"es mi edad"
```

PRINT c;p

ídem con artículos de color c.

```
PRINT INK 3;5; BRIGHT 1;" mujeres bonitas ";
INVERSE 1;"de EPSI"; FLASH 1;" le","saludan"
```

RANDOMIZE m

en el teclado, se lee RAND.

Asigna el valor m a la variable del sistema SEED. Ésta se utiliza para generar el próximo número aleatorio con la función RND. Si m = 0, que es el valor por defecto, SEED toma, en este momento, el valor de la variable del sistema FRAMES.

```
RANDOMIZE 2
PRINT RND
RANDOMIZE 2
PRINT RND
```

proporciona los 2 mismos números

```
RANDOMIZE
PRINT RND
RANDOMIZE
PRINT RND
```

proporciona dos números distintos.

READ v1,v2,...

Asigna a las variables v1, v2,... los valores situados en una lista DATA.

```
100 DATA 100,50,"buenos días"
110 READ a,b,a$
120 PRINT a,b,a$
```

haga RUN.

REM f

f puede ser lo que se quiera: comentarios, explicaciones, etc...

REM no posee ninguna influencia en el desarrollo del programa. No puede existir ninguna declaración, en la misma línea, después de REM, ya que REM puede contener el signo :

Por el contrario, REM puede encontrarse al final de la línea.

```
REM  Programa 2 : PRINT 2
```

no actúa, pero sí

```
PRINT 2 : REM  Programa 2
```

RESTORE m

reinstaura el apuntador DATA en la línea m. Este apuntador indica los datos para READ. Si la línea m no contiene declaraciones DATA, el apuntador DATA se posiciona sobre la declaración DATA siguiente. Si m = 0, valor por defecto, el apuntador DATA retorna a la 1.ª declaración DATA del programa.

```
10 DATA 10, 20, 30
20 DATA 1,2,3
RUN
READ a,b
PRINT a,b
RESTORE 15
READ a,b
PRINT a,b
RESTORE
READ a,b,c
PRINT c
```

RETURN

la pila GO SUB contiene el número de línea en la que se ha declarado la instrucción GO SUB. RETURN hace retornar a la línea siguiente.

RUN m

ejecuta CLEAR, después GO TO m
por defecto, m = 0.
La instrucción SAVE salva informaciones del ordenador hacia el grabador.
Al igual que LOAD, es una instrucción importante que permite muchas combinaciones.

SAVE "f"

salva el programa y las variables asociadas de nombre f. Éste consta de 1 a 10 caracteres como máximo. Al contrario de LOAD, que permite " ", SAVE exige un nombre f.

```
SAVE "p"
```

SAVE "f" LINE m

como SAVE "f", pero cuando vuelve a cargarse con LOAD "f", tan pronto como termina la carga, se produce un salto automático a la línea m.
RUN no es necesario, ya que, por otra parte, destruiría las variables.

```
SAVE "f" LINE 1
```

hace funcionar automáticamente su programa, en cuanto termina la carga.

SAVE "f" DATA a()

salva la matriz numérica a de nombre f.

SAVE "f" DATA a$

salva la matriz de caracteres a$

SAVE "f" CODE m,n

salva n octetos partiendo de la dirección m

SAVE "f" SCREEN$

salva el fichero de visualización. Es el mismo mandato que:

```
SAVE "f" CODE 16384,6912
```

Todos los SAVE "f" salvan informaciones sobre cassette, dándoles el nombre f.

STOP

para el programa.

VERIFY "f"
mismas posibilidades que LOAD.

VERIFY "f" DATA a()
después de haber salvado datos sobre cassette

VERIFY "f" DATA a$()
con la ayuda del mandato SAVE, se utiliza

VERIFY "f" CODE m,n
VERIFY para la verificación. VERIFY

VERIFY "f" CODE m
funciona como LOAD, salvo que los datos

VERIFY "f" CODE
no se cargan en el ordenador,

VERIFY "f" SCREEN$
aunque se comparan con los que ya están.

Anexo 2
RECAPITULACIÓN DE LAS FUNCIONES

Los parámetros de las funciones tienen los significados siguientes:

x,y	representa un número, una variable numérica o una expresión entre paréntesis
f,g	representa una cadena o una variable de cadena entre comillas
r	representa un número expresado en radianes
a	representa una letra
op	representa el operador utilizado

Para las funciones trigonométricas, se utiliza el radián como unidad de ángulo.
Para pasar de radianes a grados, se procede como sigue:

r radianes = r * 180/PI grados

y para pasar de grados a radianes, se convierten

x grados = x * PI/180 radianes

ABS x

proporciona el valor absoluto de x; así pues, convierte en positivo todo x negativo

```
LET a=-3
PRINT ABS a
```

visualiza 3

ACS x

proporciona el arco en radianes, cuyo coseno es x; por consiguiente, x debe estar comprendido entre –1 y 1

```
PRINT ACS -1
```

visualiza 3.1415927, es decir PI

AND x op y

operación Y sobre 2 operandos (no corresponde al Y lógico).
da x si y es distinto de 0
da " 0 si y = 0

"f" op x

da f, si x es distinto de 0
da " " si x = 0

ASN x

facilita el arco, en radianes, cuyo seno es x; así pues, x debe estar comprendido entre –1 y 1

```
PRINT ASN 0
```

AT

AT no es una función, sino un separador de artículos PRINT, utilizado con determinadas instrucciones.

ATN x

proporciona el arco, en radianes, cuya tangente es x

```
PRINT ATN 2
```

ATTR (x,y)

facilita los atributos de la posición PRINT (x,y), de donde:

x debe estar comprendido entre 0 y 23
y debe estar comprendido entre 0 y 31

Los atributos dados son el parpadeo, el brillo, el color papel y el color tinta, codificados de esta manera:

0 a 7 = color tinta (carácter)
(0 a 7) * 8 = color papel (fondo)
64 = brillo resaltado
128 = parpadeo

```
PRINT ATTR (0,0)
```

visualiza 56 donde el papel es blanco (7 * 8), la tinta es negra (0), el brillo es normal (0) y no hay parpadeo (0).

Introduzca la secuencia siguiente:

```
CLS
PRINT FLASH 1; ATTR (0,0)
PRINT ATTR (0,0)
PRINT ATTR (0,1)
PRINT ATTR (0,2)
```

BIN

BIN no es una función. BIN permite introducir un número más pequeño de 256 en numeración binaria.

```
LET a=BIN 10000000
PRINT a
```

visualiza 128

Esto facilita la construcción de caracteres gráficos definidos por el usuario, por lo menos en

determinados casos (véase lo dicho en este libro). Usted puede efectuar igualmente conversiones binario-decimal.
El número binario debe constar de 8 cifras 0 y/o 1. Pueden ignorarse las primera cifras 0.

```
BIN 00001000 y BIN 1000  son idénticos
LET a= BIN 1000
PRINT a
LET a= BIN 00001000
PRINT a
```

CHR$ x

facilita el carácter cuyo código es x.
véase el anexo 3 para los códigos de carácter, de los que algunos, además, corresponden a los códigos ASCII.
x debe estar comprendido entre 0 y 255.

```
PRINT CHR$ 50
PRINT CHR$ 255
```

Nota Bene: determinados códigos, no visualizables, se visualizarán por un ?.

CODE "f"

proporciona el código del primer carácter de la cadena f.

```
PRINT CODE "Adiós"
```

COS r

proporciona el coseno del ángulo r.

```
PRINT COS PI
```

E o e

no es una función, pero permite introducir un número bajo forma científica, forma bajo la que se pone automáticamente el ordenador cuando un número es demasiado grande o demasiado pequeño.

```
LET a=4e4
PRINT a
```

EXP x

proporciona la exponencial e^x
Se obtiene un valor aproximado del número transcendente e, haciendo:

```
LET e=EXP 1
PRINT e        visualiza 2.7182818
```

FN a(x,y)

realiza la función de nombre a, que ha sido definida en un programa.
Los paréntesis pueden contener una o varias variables, en número igual a las contenidas en la definición de la función por DEF FN.

```
10 DEF FN t(x)=SIN x/COS x
20 DEF FN p(x,y)=SQR (x*x+y*y)
```

FN a$(f,g)

ídem para las cadenas

INx

realiza la entrada de una boca x (periférico) al nivel del microprocesador.
(carga BC con el valor x, después hace IN A, (C) en terminología de ensamblaje de códigos máquina).

INKEY$

capta un carácter tecleado.

Se utiliza en un programa para registrar una entrada durante la ejecución del programa y orientar la continuación.

Introduzca el programa siguiente:

```
10 LET a$=INKEY$
20 PRINT a$
30 IF INKEY$< >"" THEN GO TO 30
40 IF INKEY$="" THEN GO TO 40
50 GO TO 10
```

y habrá transformado su Spectrum en una máquina de escribir.

Haga BREAK para detener este programa.

INT x

proporciona la parte entera de un número, redondeándolo por abajo.

```
PRINT INT 23.67
PRINT INT -12.23
```

LEN "f"

proporciona la longitud de la cadena f (su número de caracteres).

```
LET a$="belleza"
PRINT LEN a$
```

LN x

proporciona el logaritmo natural, de base e, del número x.
x no puede ser nulo ni negativo.

```
PRINT LN 2
```

Para obtener el logaritmo de x, en base 10, el que llamamos LOG, efectúe la operación LN x/LN 10, que es igual a LOG x.
Para obtener LOG 100, haga:

```
PRINT LN 100/LN 10
```

Esto le permite hacer tablas de logaritmos de base 10.

NOT x

da 0, si x es distinto de 0
da 1, si x es igual a 0

```
PRINT NOT 4
PRINT NOT 0
```

OR x op y

da 1, si y es distinto de 0
dda x, si y es igual a 0

```
PRINT 45 OR 45
PRINT 45 OR 0
```

PEEK x

proporciona el contenido de la dirección de memoria x.
x debe ser inferior a 65536.

```
PRINT PEEK 100
```
visualiza 255

PI

no es una función, es el valor de PI. Este número transcendente se utiliza tan a menudo que Sinclair ha reservado una palabra clave para obtener su valor.

```
PRINT PI
```

POINT (x,y)

da 1, si el pixel (x,y) está visualizado
da 0, si el pixel (x,y) no está visualizado
x debe estar comprendido entre 0 y 255
y debe estar comprendido entre 0 y 175

```
PLOT 128,88
PRINT POINT (128,88)
PRINT POINT (129,88)
```

RND

genera una secuencia de números aleatorios, comprendidos entre 0 y 1; se excluye 1.

```
PRINT RND
PRINT RND
PRINT RND
```

Si usted desea cambiar la secuencia de estos números, ya que en la conexión es siempre la misma, haga RANDOMIZE previamente. Usted entra, en este momento, en otro lugar de la secuencia.
Si desea generar números aleatorios enteros, comprendidos entre 0 y 9, haga

```
10 PRINT INT (RND*10)
20 GO TO 10
```

Puesto que RND genera un número aleatorio, con hasta 8 decimales, situado entre 0 y <1; si se multiplica RND por n, el número generado se situará entre 0 y <n. Además, si se desea un número entero, se utiliza la función INT, que elimina la parte decimal. Así, PRINT INT (RND * n) visualiza un número entero comprendido entre 0 y n-1.

Por otra parte, para visualizar un número entero comprendido entre 128 y 143, debe generarse un número comprendido entre 128 + 0 y 128 + 15, de donde n – 1 = 15 y n = 16, viene:

```
10 PRINT INT (RND*16)+128
20 GO TO 10
```

o bien

```
10 LET a=INT (RND*16)+128
20 PRINT CHR$ a;
30 GO TO 10
```

y la pantalla se llena de caracteres gráficos de código 128 a 143 de una forma aleatoria. Para visualizar únicamente números pares (por ejemplo), se multiplica por 2. Los límites cambian. Visualicemos números pares comprendidos entre 30 y 70 (comprendido el 70).

```
10 PRINT 2*INT (RND*21)+30
20 GO TO 10
```

el límite superior es (n – 1) * 2 = 40, de donde n = 21

SCREEN$ (x,y)

proporciona el carácter que está visualizado en la posición PRINT (x,y); así pues, x debe estar comprendido entre 0 y 23, y entre 0 y 31.

```
10 LET a$="Vd. es bonita"
20 PRINT a$
30 FOR n=1 TO LEN a$
40 PRINT SCREEN$ (0,n)
50 NEXT n
```

El texto de a$ se visualiza igualmente de forma vertical en la 1.ª columna. Usted puede, de todas formas, volver a visualizar este texto en cualquier lugar, cambiando la línea 40

SGN x

facilita el signo de x, o sea:
–1 para un número negativo
0 para un número nulo

1 para un número positivo

Esta función se aplica preferentemente a una variable, ya que usted no pedirá nunca al ordenador el signo de 23.

```
LET a=304
PRINT SGN a
```

SIN r

proporciona el seno del ángulo r.

```
10 FOR n=0 TO 90 STEP 5
20 PRINT n,SIN (n*PI/180)
30 NEXT n
```

Este pequeño programa visualiza el seno de n, expresado en grados, para n que va de 0 a 90, de 5 en 5 grados.

SQR x

facilita la raíz cuadrada de x, para x positivo.

STR$ x

transforma el número x en cadena

```
LET a=1234
LET a$=STR$ a
PRINT a$
```

o bien

```
LET bs=STRs 2
PRINT bs
```

Esta función es útil para economizar octetos de memoria. Cualquier número necesita 6 octetos de más, todas las veces que aparezca en una declaración; x = 1 necesita 6 octetos de memoria más que x$ = "1" (al menos en el nivel del número).

Para utilizar el valor de una cadena, véase VAL.

TAB

TAB no es una función, es un separador de artículos PRINT, al igual que AT.

TAN r

proporciona la tangente del ángulo r.

```
PRINT TAN (PI/2)
```

facilita teóricamente el infinito; el ordenador responde: número demasiado grande.

TO

TO no es realmente una función, sino un separador utilizado en los circuitos FOR (vea el ejemplo dado para la función SIN) y para dividir una cadena. En este último caso, esta pequeña palabra es muy potente.

```
LET a$="buenos días"
PRINT a$(TO 3)
PRINT a$(4 TO)
PRINT a$(2 TO 3)
```

USR x

llama a la rutina en código máquina, cuya dirección de inicio es x, y la ejecuta. Al retorno, el resultado es el contenido de BC (al nivel del microprocesador). Si usted ha introducido los C. M. en las direcciones 32000 y sec., haga

```
PRINT USR 32000
```

Si usted hace

```
PRINT USR 0
```

se reinicializa todo el sistema. 0 es la dirección de inicio, cuando se conecta el ordenador. Todas las informaciones situadas después de RAMTOP: códigos máquina, gráficos definidos, etc. se pierden. Si usted desea conservar estas informaciones, haga

```
NEW
```

USR "f"

proporciona la dirección de la matriz de bits del gráfico definido por el usuario correspondiente a f. f es una letra simple, comprendida entre a y u, ya que solamente pueden definirse 21 gráficos.

Encontrará ejemplos dentro del libro. Para definir más de 21 gráficos, véase el anexo 5.

VAL "f"

transforma la cadena f en una expresión numérica, siempre que f contenga una expresión válida.

```
LET a$="bueno"
PRINT VAL a$
```

da un mensaje de error, pero esto

```
LET bueno=2
PRINT VAL a$       visualiza 2
```

VAL es algunas veces útil para sustituir DEF FN

```
DEF FN
```

Si usted desea captar un número aleatorio comprendido entre 0 y 9, haga

```
LET a$="INT (RND*10)"
PRINT VAL a$
```

dará, cada vez que se utilice, este número aleatorio.

VAL$ "f"

esta función no es corriente y es muy poco utilizada. Evalúa f para devolver una expresión de cadena, después de haber quitado las comillas.

```
LET a$="hola"
PRINT VAL$ "a$"
PRINT VAL$ """a$"""
```

no es una función propiamente dicha; se utiliza para permitir varias instrucciones en la misma línea de programa o de mandato.

```
LET a=5 : PRINT a
```

– x

menos unario (que actúa sobre un operando). Para introducir números negativos.

+ x op y
"f" op "g"

adición de 2 números
encadenamiento de 2 cadenas

```
LET a$="cabeza"
PRINT a$+"-a-"+a$
```

– x op y

sustracción de 2 números

*** x op y**

multiplicación de 2 números

/ x op y

división de 2 números

↑ x op y

elevación de x a la potencia de y.

Si x es negativo, el resultado no será matemáticamente exacto y podrá ser positivo, negativo o fraccionario

```
PRINT 2↑15
PRINT 2↑-2
PRINT 2↑.5
```

este último es el mismo que

```
PRINT SQR 2
```

para efectuar una raíz cúbica

```
PRINT 8↑(1/3)
```

los operadores de comparación

=	igual a
>	mayor que
< **x op y**	menor que
< = **"f" op "g"**	menor que o igual a
> =	mayor que o igual a
< >	distinto de

los 2 operandos deben ser del mismo tipo.
El resultado de la comparación es
1 si es verdadera
0 si es falsa
La comparación de las cadenas actúa sobre el alfabeto y no sobre la longitud.

```
PRINT 5>3
PRINT 5=3
PRINT "amor"<"bueno"
```

Las funciones y operadores de arriba tienen sus prioridades, unos sobre otros, en la evaluación de una expresión. Son las siguientes:

operador	**prioridad**
índice y partición	12
todas las funciones excepto NOT	11
y – unario	
↑	10
– unario	9
* y /	8
+ y – (sobre 2 operandos)	6
de comparación	5
NOT	4
AND	3
OR	2

Los paréntesis cambian las prioridades, ya que el ordenador evalúa primero el contenido de los paréntesis.
Teniendo en cuenta las prioridades, hemos introducido paréntesis en algunos ejemplos que ilustran las funciones.
PRINT 8 ↑ 1/3 visualiza 2.6666667
el ordenador efectúa primero la exponenciación, después la división.
PRINT 8 ↑ (1/3) visualiza 2
el ordenador efectúa la división, después la exponenciación.

Anexo 3

CÓDIGOS DE LOS CARACTERES

Los caracteres disponibles están codificados sobre un octeto, es decir de 0 a 255.
He aquí los 32 primeros caracteres. Son propios del Spectrum.

código	**carácter**
0 a 5	no utilizados
6	PRINT coma
7	EDIT
8	cursor izquierda
9	cursor derecha
10	cursor abajo
11	cursor arriba
12	DELETE
13	ENTER
14	número (seguido de 5 octetos que representan el número en coma flotante)
15	no utilizado
16	mandato INK
17	mandato PAPER
18	mandato FLASH
19	mandato BRIGHT
20	mandato INVERSE
21	mandato OVER
22	mandato AT
23	mandato TAB
24 a 31	no utilizados

La utilización de estos códigos no es frecuente en programación Basic corriente. Recarga el mandato y lo vuelve más confuso. ¿Qué opina usted de

```
PRINT 8;CHR$ 6;10
```

en lugar de

```
PRINT 8,10
```

con el mismo resultado?

o bien de

```
PRINT CHR$ 18;CHR$ 1;3
```

en lugar de

```
PRINT FLASH 1;3
```

Fíjese en que CHR$ 1 es el parámetro 1 del mandato CHR$ 18, y no el carácter de código 1

```
PRINT CHR$ 1        ¿visualiza?
```

Esto

```
PRINT 7;CHR$ 8; CHR$ 21; CHR$ 1; CHR$ 95
```

para visualizar un 7 subrayado
El equivalente sería:

```
PRINT AT 0,0;7;AT 0,0;OVER 1;" "
```

Los caracteres de códigos 32 a 127 son directamente visualizables. Al contrario de ZX81, cuyos caracteres visualizables tenían sus códigos personales, los del Spectrum corresponden igualmente a los códigos ASCII, con excepción de los signos 'libra' y 'copyright', códigos 96 y 127, los cuales son propios del Spectrum. Casi todos los ordenadores utilizan los códigos ASCII para los caracteres visualizables. El código 65 representa A en todos los sistemas que siguen el código ASCII.

código	carácter	código	carácter
32	espacio	46	.
33	!	47	/
34	”	48	0
35	#	49	1
36	$ (signo dólar)	50	2
37	%	51	3
38	&	52	4
39	’	53	5
40	(	54	6
41	)	55	7
42	*	56	8
43	+	57	9
44	,	58	:
45	–	59	;

código	carácter	código	carácter
60	<	94	↑
61	=	95	–
62	>	96	£ (signo libra esterlina)
63	?	97	a
64	@	98	b
65	A	99	c
66	B	100	d
67	C	101	e
68	D	102	f
69	E	103	g
70	F	104	h
71	G	105	i
72	H	106	j
73	I	107	k
74	J	108	l
75	K	109	m
76	L	110	n
77	M	111	o
78	N	112	p
79	O	113	q
80	P	114	r
81	Q	115	s
82	R	116	t
83	S	117	u
84	T	118	v
85	U	119	w
86	V	120	x
87	W	121	y
88	X	122	z
89	Y	123	{
90	Z	124	|
91	[	125	}
92	/	126	~
93	]	127	©

Todos estos caracteres están formados por una matriz de puntos. Cada matriz contiene 8 octetos. Para encontrar el comienzo de cada matriz, usted hace 15360 + 8 veces el código del carácter referido. De esta manera, la matriz del carácter 'espacio' empieza en 15360 + 8 * 32, o sea 15616. Evidentemente, los 8 octetos están a 0. La matriz de A empieza en 15880, etc.

Los caracteres de códigos 128 a 143 son caracteres gráficos predefinidos.

código	carácter	código	carácter
128	□	136	▖
129	▝	137	▞
130	▘	138	▌
131	▀	139	▛
132	▗	140	▄
133	▐	141	▟
134	▚	142	▙
135	▜	143	█

Los caracteres de código 144 a 164 son caracteres gráficos a definir por el usuario. En la inicialización, los códigos 144 a 164 contienen las letras de la A a la U.

Los caracteres de códigos 165 a 255 (véase manual) representan las funciones y las instrucciones específicas del Spectrum. Éstas son palabras clave, y si usted las visualiza, se extenderán sobre sus longitudes.

```
PRINT CHR$ 249
```

visualiza RANDOMIZE (con un espacio delante y otro detrás).

Anexo 4

INFORME DE ERRORES

El informe que aparece en la zona de edición (línea 23) comprende:

- una cifra o una letra de identificación
- un texto explicativo breve terminado por una coma
- dos números separados por 2 puntos. El primer número indica el número de línea que ha provocado la visualización del informe, y el 2.° número indica qué instrucción, en esta línea, es la responsable. Cuando se trata de un mandato, el 1.er número es 0.

El informe aparece cuando el ordenador se para por una razón u otra.

0 OK,

todo va bien. Por ejemplo:

```
PRINT 1                  da
0 OK, 0:1
PRINT 1 : PRINT 2        da
0 OK, 0:2
```

1 NEXT without FOR,

se ha utilizado la instrucción NEXT sin la instrucción previa FOR.

```
LET a=2
NEXT a
```

2 Variable not found,

variable no encontrada

```
CLEAR
PRINT a
```

3 Subscript wrong,

índice erróneo

```
DIM a(10)
PRINT a(11)
```

4 Out of memory,

no hay bastante espacio en la memoria

```
CLEAR 24000
DIM a(100)
```

5 Out of screen,

fuera de la pantalla

```
PRINT AT 22,2;
```

6 Number too big,

número demasiado grande;
de hecho, más grande de alrededor de 10^{38}

```
PRINT 1/0
```

7 RETURN without GO SUB,

la instrucción RETURN ha sido utilizada sin un GO SUB previo.

```
RETURN
```

8 End of file,

fin de fichero
reservado a los periféricos tales como microdrive, etc...

9 STOP statement,

declaración STOP

```
STOP
```

A Invalid argument,

argumento de una función no válido

```
PRINT SQR -4
```

B Integer out of range,
entero fuera del rango previsto como argumento

```
PLOT 256,2
```

C Nonsense in Basic,
no tiene sentido en Basic

```
PRINT VAL "4t"
```

D BREAK – CONT repeats,
paro provocado por BREAK, la instrucción CONT vuelve a tomar la ejecución.

```
10 PRINT 3
20 GO TO 10
```

pulse BREAK

E Out of DATA,
aparece cuando se intenta READ, cuando se han leído todos los datos de las instrucciones DATA.

```
10 DATA 10,20,30
20 READ a,b,c,d
```

F Invalid file name,
nombre de fichero no válido,

```
SAVE ""        o SAVE "programa 12"
```

un nombre vacío o de más de 10 caracteres no es aceptado por la instrucción SAVE.

G No room for line,
no hay más espacio en la memoria para aceptar una nueva línea de programa,

H STOP in INPUT,
STOP como entrada de un INPUT

```
10 INPUT "Introduzca su edad",a
```

Usted pide a su vecina que responda y ella introduce STOP

I FOR without NEXT,
FOR sin instrucción NEXT
Usted quiere visualizar los números de 9 a 0 con el programa:

```
10 FOR n=9 TO 0 STEP -1
20 PRINT n
30 NEXT n
```

pero ha olvidado el STEP y el NEXT, de donde

```
10 FOR n=9 TO 0
20 PRINT n
```

RUN visualiza el informe I
Es muy poco probable que le suceda esto.

J Invalid I/O device,
dispositivo de entrada/salida no válido, reservado a los periféricos: microdrive, etc.

K Invalid colour,
color no válido,

```
INK 11
```

L BREAK into program,
Se ha tecleado BREAK durante la ejecución de un programa,

```
10 PRINT "//";
20 PRINT "\";
30 GO TO 10
```

Si pulsa BREAK cuando la pantalla está llena y el ordenador visualiza scroll?, obtendrá el error D.

M RAMTOP no good,

el número especificado para RAMTOP no es correcto

```
CLEAR 23800
```

N Statement Lost,

declaración perdida;
no ocurre casi nunca, el ordenador no pierde sus declaraciones (línea de programa) tan rápido.

O Invalid Stream,

corriente no válida;
puede tratarse de la corriente de entrada.
Reservado a los periféricos tales como microdrive, etc.

P FN without DEF,

utilización de FN sin que haya DEF FN

```
PRINT FN d(5)
```

Q Parameter error,

error de parámetro en una función FN

```
10 DEF FN a(x)=x↑3
20 PRINT FN a(5)
30 PRINT FN a("5")
```

R Tape loading error,

error de carga de la cinta;
cuando el programa ha sido encontrado en la cinta, no podrá ser ni cargado ni verificado ni mezclado, por una razón no definida.

Anexo 5

LAS VARIABLES DEL SISTEMA

Las variables del sistema son variables utilizadas por el Spectrum para el buen funcionamiento de sus operaciones. Están situadas en los octetos de memoria, cuyas direcciones van de 23552 a 23733.

Todas las variables del sistema pueden leerse en cualquier momento. Prácticamente, usted puede leer una variable del sistema, efectuar una operación y después volver a leer esta variable, para comprender cómo la utiliza el ordenador. Algunas no pueden utilizarse de ningún modo, en detrimento de la "configuración" del ordenador. Otras solamente tienen una acción transitoria; su utilización no sería, pues, útil. Las que pueden modificarse a voluntad se examinarán aquí. Su manipulación ocasionará resultados divertidos y algunas veces asombrosos.

Las variables del sistema que manejan un número más pequeño que 256, ocupan solamente un octeto de memoria. Se utiliza la instrucción POKE para ocupar este octeto:

```
POKE 23681,100
```

pone el número 100 en el octeto de dirección 23681.
Se utiliza la función PEEK para leer este octeto. Ya que PEEK es una función, insertamos antes una instrucción:

```
PRINT PEEK 23681
```

visualiza 100 en la pantalla.

Las variables del sistema que manejan un número más pequeño que 65536 ocupan dos octetos de memoria, primero el menos significativo y, a continuación, el más significativo. Esto es de uso general, con una sola excepción: el número de línea de un programa.

Por ejemplo, para almacenar 40000 en dos octetos de memoria, hagamos 40000 / 256, es decir 156.25. El octeto de mayor peso (el más significativo) será 156.

Tiene 156 * 256, o sea 39936. El octeto de menos peso (el menos significativo) es 40000 – 39936, o sea 64.

Para cargar 40000, hagamos:

```
POKE 23728,64
POKE 23729,156
```

y, para leer:

```
PRINT PEEK 23728+256*PEEK 23729
```

Antes de examinar las variables del sistema disponibles, intentemos comprender cómo almacena el Spectrum las líneas de programas y las variables de éste, con la ayuda del programa siguiente:

```
 5 LET A=22
10 LET B=23627
15 LET C=PEEK B+256*PEEK (B+1)
20 FOR N=0 TO 19
25 PRINT C+N,PEEK (C+N)
30 NEXT N
```

La dirección de las variables del programa se encuentra en la variable del sistema. VARS de direcciones 23627/8.

La dirección del programa se encuentra en la variable del sistema. PROG de direcciones 23635/6.

Hagamos funcionar el programa de arriba; obtendremos:

		observaciones	
23870	97	(	código de a
23871	0	(	5 octetos que
23872	0	(	representan
23873	22	(	22
23874	0	(	en coma
23875	0	(	flotante
23876	98	(	código de b
23877	0	(	5 octetos que
23878	0	(	representan
23879	75	(	23627
23880	92	(	en coma
23881	0	(	flotante

La variable siguiente es c de código 99.

La variable siguiente es n de código 238 – 128, es decir 110.

Las variables cuyo nombre se compone de una letra confunden las mayúsculas y las minúsculas. La variable A es idéntica a la variable a.

Para una variable de control de circuito (siempre una sola letra), el ordenador añade 128 a su código de minúscula, para identificarlo.

En el programa precedente, hagamos:

```
10 LET B=23635
```

para examinar el almacenamiento del programa:

			observaciones
23755	0	(	número de línea =
23756	5	(	0 * 256 + 5 * 1 = 5
23757	12	(	número de caracteres =
23758	0	(	12 * 1 + 0 * 256 = 12
23759	241	(	código de LET
23760	65	(	código de A
23761	61	(	código de =
23762	50	(	código de 2
23763	50	(	código de 2
23764	14	(	código de 'número'
23765	0	(	seguido de 5 octetos
23766	0	(	para representar
23767	22	(	el número en
23768	0	(	coma
23769	0	(	flotante
23770	13	(	código de 'ENTER'

Los 2 primeros octetos dan el número de línea (primero el octeto más significativo).

Los 2 octetos siguientes dan la longitud de la declaración (primero el octeto menos significativo). Aprendemos que hay 12 octetos a partir de 23758. La línea 5 está almacenada de 23755 a 23770. Todas las veces que hay un número en una declaración, el ordenador añade 6 octetos: el octeto de código 14 más 5 octetos que representan el número en coma flotante. Estos 5 últimos octetos están además reproducidos en la zona de variables, como primer valor asignado a A, cuando el programa se ejecuta. Haga:

```
RUN
PRINT A
```
visualiza 22

después haga:

```
CLEAR
PRINT A
```
visualiza un mensaje de error

Examinemos las variables del sistema modificables.

23561 REPDEL

tiempo, en 50.° de segundo, durante el cual una tecla debe permanecer pulsada para empezar su repetición.

```
PRINT PEEK 23561
```
visualiza 35

haga:

```
POKE 23561,5
```

el teclado es difícil de manejar, haga

```
POKE 23561,35
```

es un buen valor

23562 REPPER

retrasa, en un 50.° de segundo, las repeticiones sucesivas de una tecla que se ha mantenido pulsada.

```
PRINT PEEK 23562
```

haga

```
POKE 23562,1
```

Deje la tecla 1, o cualquier otra, pulsada.
Esto es práctico para llenar una línea con un mismo carácter, o para borrar una línea. Haga

`POKE 23562,5` visualiza 5

23606/7 CHARS

contiene la dirección de las matrices del juego de caracteres, código 32 a código 127, menos 256.

¿Por qué –256? Porque los 32 primeros caracteres, códigos 0 a 31, no son visualizables, y cada matriz de carácter está formada por 8 octetos, conteniendo cada uno de ellos una combinación de 8 pixels (véase anexo 3).

¿Qué contiene CHARS?

```
PRINT PEEK 23606 + 256* PEEK 23607
```

visualiza 15360.

El ordenador, para encontrar los 8 octetos de la letra A, código 65, ejecuta 15360 + 65 veces 8, es decir 15880. Los 8 octetos que forman A están en las direcciones 15880 a 15887. Ponga estos octetos en binario, uno debajo del otro, y obtendrá el dibujo de A en una rejilla de 8 veces 8 casillas, o pixels.

Si usted desea volver a definir todo el juego de caracteres, deberá volver a definir 8 veces 96, es decir 768 octetos, e introducirlos, por ejemplo, en las direcciones 31001 a 31768, después de haber situado RAMTOP más abajo de 31001. Cuando el trabajo ha terminado, se reserva en cassette; ponga 30745 en la variable del sistema CHARS. Para dibujar e introducir sus nuevos caracteres, proceda del mismo modo que con los caracteres GDU (véase dentro del libro).

Divirtámonos un instante con esta variable del sistema. Veamos su contenido:

```
PRINT PEEK 23606      visualiza 0
PRINT PEEK 23607      visualiza 60
```

esto es lo que esperábamos, ya que 60 * 256 = 15360.

Escojamos un juego de caracteres divertido, efectuando

```
POKE 23607,0
```

¿Comprende usted el informe?

No se asuste, su ordenador funciona normalmente, pero con una visualización misteriosa, con caracteres de matrices fantasiosas. Si ha dejado usted el programa de 6 líneas del principio de este capítulo, teclee ENTER: listará su programa en caracteres misteriosos y, debajo, la K parpadeante.

Haga

```
RUN ENTER
```

¡¡¡Y la pantalla visualiza la ejecución del programa!!! haga:

```
POKE 23607,60
```

(si cree que se ha equivocado, antes de teclear ENTER, utilice DELETE para volver al principio de la línea editada.)

La visualización vuelve a ser normal.

También es posible hacer que CHARS apunte a un carácter más alto, así pues 8 octetos de más:

```
POKE 23606,8        (octeto bajo de la dirección)
```

Observe el informe, el listado, la ejecución. Los 5 se convierten en 6, las L en M, etc. Lo que es curioso, es tomar 4 octetos de A y 4 de B, para sustituir A; haga

```
POKE 23606,4
```

divertido, ¿no?

23608 RASP

duración de sonido que el ordenador emite para advertirle que la memoria está llena y que rechaza toda nueva línea. Es poco probable que ya haya oído este sonido. Haga

```
NEW
CLEAR 23860
PRINT "aaaa
```

si usted intenta introducir un 5.° a, oirá el sonido.

Disculpe pero, para continuar, hay que cortar la corriente y volver a ponerla; quería que usted oyera el sonido una vez por lo menos.

23609 PIP

duración del clic cuando se pulsa una tecla.

```
POKE 23609,30
```

y usted obtiene un clic más largo cuando pulsa una tecla. Es más fácil para programar. Haga este mandato cada vez que utilice el Spectrum.

23610 ERR–NR

es el número del informe menos uno.

Para tener el informe J, cargue 19 – 1

```
POKE 23610,18
```

No se preocupe, el ordenador ha vuelto a poner esta variable del sistema para tener el informe 0, es decir 0 – 1, ¿adivina qué número?

```
PRINT PEEK 23610
```

Para tener el informe R, cargue 27 – 1

```
POKE 23610,26
```

a continuación, pruebe:

```
POKE 23610,27
POKE 23610,28
POKE 23610,29
```

Su curiosidad le hará hacer

```
POKE 23610,30
```

se decepcionará, el mensaje de error se resume en muy poca cosa.

23618/9 NEWPPC

contiene el número de la línea hacia la cual desea usted saltar.

23620 NSPPC

contiene el número de declaración, en la línea que usted ha introducido en NEWPPC, hacia la cual desea saltar.

Haga el programa:

```
10 PRINT "Spectrum"
20 POKE 23618,10 :
   POKE 23620,1
```

pulse RUN

La línea 20 le envía constantemente a la línea 10, 1.ª declaración (sólo hay una, pero tiene que especificarse en 23620).

La línea 20 es totalmente incomprensible para los principiantes, mientras que GO TO 10...

23621/2 PPC

contiene el número de línea de la declaración que se ejecuta.

23623 SUBPPC

contiene el número de declaración en la línea que se ejecuta.

Es inútil llenar estas dos variables del sistema, cambian en cuanto usted pulsa ENTER.

23624 BORDCR

contiene el color de los márgenes multiplicado por 8.
¿Desea usted los márgenes en color cyan? Haga POKE 23624,40 y usted verá que la zona de edición está igualmente en cyan. De hecho, esta variable del sistema actúa de esta manera: introduciendo
0 a 7 = color tinta zona de edición
(0 a 7) * 8 = color papel zona de edición y color de los márgenes.
64 = asigna el brillo de la zona de edición
128 = asigna el parpadeo de la zona de edición
Estos números son acumulativos.
Usted desea tinta verde 4
Usted desea papel blanco 56 (también para los márgenes)
Usted desea un brillo resaltado 64
Usted desea un parpadeo 128
de donde,

```
POKE 23624,252
```

vuelva a la situación normal, realizando

```
POKE 23624,56
```

23625/6 E–PPC

contiene el número línea sobre la cual apunta el cursor de línea.

```
PRINT PEEK 23625
POKE 23625,30
```

visualiza 50

y el cursor de línea apunta sobre la línea 30. Es muy práctico para llevar el cursor a la línea que se desea editar, para modificarla. No olvide que, si su n.° de línea supera 255, es necesario hacer intervenir los 2 octetos. Es decir N, un n.° de línea sobre la cual usted desea posicionar el cursor; haga:

```
POKE 23625,N-INT (N/256)*256
POKE 23626,INT (N/256)
```

23643/4 K–CUR

proporciona la dirección del cursor de la línea editada.

Cuando se introduce un mandato, la línea editada aumenta con la entrada de caracteres, y el cursor se desplaza. La dirección del cursor varía constantemente y esta variable del sistema se actualiza continuamente por el ordenador. Su cumplimentación no sirve para nada. Para tener una idea de su funcionamiento, haga:

```
NEW
PRINT PEEK 23641+256*PEEK 23642
```

visualiza 23756

La variable del sistema 23641/2,E–LINE, proporciona la dirección del inicio de la línea editada; en este momento sabemos que es 23756. Haga el mandato

```
PRINT PEEK 23643+256*PEEK 23644
```

se obtiene 23786.

Si el inicio de la línea editada es 23756, 23786 indica 31 posiciones más lejos: era el sitio del cursor L parpadeante, en el momento en que usted ha pulsado ENTER.

Si usted vuelve a hacer el mandato de arriba, añadiendo un espacio al final, obtendrá 23787.

23647/8 X–PTR

proporciona la dirección del carácter situado después del marcador ? Este marcador señala un error de sintaxis en la línea editada.

23658 FLAGS2

este octeto se compone de 8 indicadores, que el Spectrum utiliza para gestionar su funcionamiento.

23660/1 S–TOP

proporciona el n.° de la línea de programa situada arriba de la pantalla, en listing automático (mandato LIST).

23662/3 OLDPPC

proporciona el n.° de la línea de programa a la que salta CONTINUE.

23664 OSPCC

si en la línea arriba definida hubiera varias declaraciones, esta variable del sistema proporciona el número de la declaración a la que salta CONTINUE.

Estas 5 últimas variables del sistema sirven para la gestión del sistema. No tienen ninguna utilidad directa.

23670/1 SEED

contiene un número guía que se utiliza para la función RND. Éste genera, a cada llamada, un número aleatorio menor que 1, efectuando cálculos sobre este número guía, después almacena en SEED un nuevo número guía que servirá para la llamada siguiente. La función RND genera, de esta manera, 65536 números aleatorios, todos distintos; después vuelve a empezar esta generación en el mismo orden. Al inicio, después de la inicialización, SEED contendrá 0. Haga:

```
10 PRINT PEEK 23670+256*PEEK 23671,
20 PRINT RND
30 GO TO 5
```

y obtendrá, por cada número guía contenido en SEED, el número aleatorio correspondiente generado por RND. El mandato RANDOMIZE fuerza la asignación de un número en SEED. Añada, por ejemplo:

```
5 RANDOMIZE 7334
```

y SEED contendrá constantemente 7334
y RND generará constantemente 0.394104
Efectuando simplemente el mandato RANDOMIZE, sin indicar ningún parámetro, el número guía que se introducirá en SEED se toma de la variable del sistema FRAMES. En el programa de arriba, haga:

```
5 RANDOMIZE
```

y vea...
En sus programas que utilicen RND, haga una declaración

```
10 RANDOMIZE
```

al inicio de éstos.
Sus números aleatorios serán imprevisibles.

23672/3/4 FRAMES

He aquí una variable del sistema de 3 octetos. Para obtener su contenido, haga:

```
PRINT PEEK 23672+256*PEEK 23673+65536*PEEK 23674
```

FRAMES se incrementa cada vez que el ordenador envía una imagen, una trama, a la TV, o sea 50 veces por segundo. Disponemos pues de un reloj dotado de una precisión de 1/50 de segundo. Utilícelo para todos los usos que pueda imaginar.
Por ejemplo, calculemos el tiempo que tarda el ordenador para realizar sus mandatos:

```
10 POKE 23672,0
20 FOR n=1 TO 100
30 NEXT n
40 PRINT 2*PEEK 23672
```

En la línea 10, ponemos a 0 el octeto de menor peso. Como este octeto va constantemente de 0 a 255, limitamos nuestro cronometraje a 256/50.°, es decir, un poco más de 5 segundos, lo que es ampliamente suficiente.
En la línea 40, multiplicamos por 2 para obtener 100.°s de segundo, lo que es más expresivo.
Haga funcionar este programa, obtendrá 42/100 de segundo para que el circuito se efectúe 100 veces.
¿Cuánto tiempo dura LET A=1?
Introduzca la línea:

```
25 LET A=1
```

Usted obtendrá 64/100 de segundo.
64/100 – 42/100 = 22/100 para efectuar 100 veces

```
LET A=1
```

Haga:

```
25 PRINT 1;
```

Se obtiene 138 – 42, es decir, 96/100 para efectuar 100 veces PRINT 1;

Para PRINT "p"; esto dura menos tiempo. Lo mismo para PRINT 0.

```
PRINT 4*.25
```

dura 224 – 42, es decir 182/100 para efectuarse 100 veces.

```
PRINT SIN PI;
```

dura aún más tiempo. Usted consigue, de esta manera, el tiempo que duran las operaciones matemáticas sobre los números.

Nota: Algunas rutinas del Spectrum y algunos mandatos, como BEEP, interrumpen el incremento de FRAMES.

23675/6 UDG

proporciona la dirección del primer gráfico definido por el usuario o, más exactamente, la dirección del primer octeto del primer gráfico. Haga:

```
PRINT PEEK 23675+256*PEEK 23676
```

el cual visualiza 32600 para un Spectrum de 16 k y 65368 para un Spectrum de 48 k.

En la inicialización, el ordenador vuelve a copiar en la zona UDG las matrices de los caracteres A a U.

Cuando usted define un nuevo carácter gráfico UDG o GDU, en castellano, (Gráfico Definido por el Usuario), sustituye una de las 21 letras A a U por su nuevo GDU.

Existen dos medios para visualizar esta letra o este GDU.

1) PRINT "A"

Este primer medio es más complejo de lo que parece: usted teclea sucesivamente:
– PRINT
– "
– paso en modalidad G
– A
– paso en modalidad L
– "
– ENTER
Hay paso en modalidad G obligatorio

2) PRINT CHR$

Este 2.° medio es más simple.
Usted obtiene los 21 GDU con los códigos 144 a 164.
Para construir tantos GDU como se desee, a pesar de que solamente se dispone de 21 mandatos para visualizarlos, los construimos por múltiplos de 21.
Supongamos la construcción de 2 juegos de 21 caracteres GDU; primero disminuimos RAMTOP, antes de introducir el programa,

```
CLEAR 31000          (64000)
NEW
```

(ponemos entre paréntesis el número a introducir para un Spectrum de 48 k).
Después, introduzca el programa:

```
  10 LET JUEGO1=9500
  20 LET JUEGO2=9510
9499 STOP
9500 LET GDU=32600 : REM (65368)
9505 GO TO 9515
9510 LET GDU=32432 : REM (65200)
9515 POKE 23675,GDU-INT (GDU/256)*256
: POKE 23676,INT (GDU/256)
9520 RETURN
```

Líneas 10 y 20: asignación de 2 variables.
Línea 9500: 32600 (65368) es la dirección del GDU inicial.
Línea 9510: 32432 (65200) es la nueva dirección de GDU, es decir, 21 veces 8 octetos menos.
Línea 9515: la dirección de GDU se pasa a la variable del sistema UDG. Para la utilización, usted hace GO SUB JUEGO1 o GO SUB JUEGO2, según desee acceder al 1.er o 2.° juego de 21 caracteres GDU. Veamos un ejemplo.
En el 1.er carácter del 1.er juego, introduzcamos un pequeño tablero de cuadrados blancos y negros de 4 pixels.
En el 1.er carácter del 2.° juego, introduzcamos 2 pequeñas líneas verticales de 2 pixels de ancho.

```
  30 DATA 51,204,51,204,51,204,51,204
  40 DATA 51,51,51,51,51,51,51,51
  50 FOR N=65368 TO 65375
: REM       32600 TO 32607   para 16k
  60 READ A : POKE N,A
  70 NEXT N
  80 FOR N=65200 TO 65207
: REM       32432 TO 32439   para 16k
  90 READ A : POKE N,A
 100 NEXT N
```

Visualicemos una línea de cada carácter GDU:

```
500 GO SUB JUEGO1
505 FOR N=0 TO 31
510 PRINT CHR$ 144;
515 NEXT N
520 GO SUB JUEGO2
525 FOR N=0 TO 31
530 PRINT CHR$ 144;
535 NEXT N
```

Haga 42 caracteres GDU e introdúzcalos de 32432 (65200) a 32767 (65535), por grupos de 8 octetos por carácter.
GO SUB JUEGO1 da acceso a 21 caracteres GDU.
GO SUB JUEGO2 da acceso al 2.° juego de 21 caracteres GDU.

23677 COORDS

Contiene la abscisa x del último pixel visualizado de coordenadas (x,y)

23678 COORDS

contiene la ordenada y del último pixel visualizado de coordenadas (x,y).

Situando números en estas 2 variables del sistema, usted modifica la posición PLOT sin utilizar esta instrucción, es decir, sin visualización de pixel.

```
POKE 23677,128
POKE 23678,88
DRAW 127,87
```

traza una línea, desde el centro de la pantalla hacia el ángulo superior derecho.

Evidentemente es más fácil hacer

```
PLOT 128,88
```

que los 2 POKE, pero con éstos usted no visualiza ningún pixel. Para ello, debería hacer en Basic

```
PLOT 128,88 : PLOT OVER 1;128,88
```

23679 P–POSN

facilita el complemento a 33 del número de columna para la impresora.

23680 PR–CC

proporciona la posición siguiente para LPRINT.

Estas dos variables del sistema son útiles para aquellos que desean imponer una columna de impresión distinta de la de visualización.

23681

no utilizada por el sistema. Haga con ella lo que le parezca.

23683 ECHO–E

contiene el número de columna bajo la forma 33-X del tampón de entrada.

23682 ECHO–E

contiene el número de línea y del tampón de entrada.

(x,y) son las coordenadas de la posición PRINT para la parte inferior de la pantalla, ya que se trata del tampón de entrada.

Estas variables del sistema son constantemente actualizadas por el Spectrum, y como la instrucción PRINT no permite, en principio, visualizar en la 24.ª línea (línea 23) y, por último, como desde que se pulsa ENTER el mandato editado se borra directamente del tampón de entrada, la lectura de estas variables del sistema da siempre los mismos números.

```
PRINT PEEK 23682          visualiza 33
PRINT PEEK 23683          visualiza 23
```

Es la posición PRINT (0,23), o bajo otra forma (33 – 0,23), es decir la l.ª columna de la 24.ª línea. De todas formas, he aquí un medio para visualizar en la 24.ª línea de la pantalla; lo que nos permitirá ver el contenido de estas variables del sistema. Este medio consiste en poner ≠ 1 como artículo PRINT. Haga el programa:

```
10 PRINT #1;"PP";PEEK 23682;"  ";PEEK 23683
20 PAUSE 4e4
```

y verá sobre la 24.ª línea:
PP31 23
o sea la posición PRINT (33 – 2,23)
La línea 20 es necesaria para evitar el informe. En la línea 10, en lugar de 2 letras P, ponga 32 letras P; verá las 32 letras P y después 1,22, o sea la posición (33 – 32,22)

23684/5 DF–CC

proporciona la dirección, en el fichero de visualización, de la posición PRINT.

Haga:

```
CLS
PRINT PEEK 23684+256*PEEK 23685
```

Usted obtiene 16384, que es el inicio del fichero de visualización.

El fichero de visualización y su gestión son particulares del Spectrum. Sin embargo, si desea

usted conocer la dirección, en el fichero de visualización, del inicio de cada línea de la pantalla, haga el programa

```
10 FOR N=0 TO 21
20 PRINT PEEK 23684+256*PEEK 23685
30 NEXT N
```

23686/7 DFCCL

da la dirección, en el fichero de visualización, de la posición PRINT, para la parte inferior de la pantalla.
Haga

```
PRINT PEEK 23686+256*PEEK 23687
```

Obtendrá 20704, que es la dirección del inicio de la 24.ª línea.

El inicio de la 23.ª línea es 20672.

23692 SCR–CT

descuenta los desplazamientos de la pantalla hacia arriba, línea por línea, cuando el ordenador visualiza la pregunta scroll?

De esta manera, sobre una respuesta favorable, el ordenador efectúa 22 desplazamientos hacia arriba para que aparezcan las 22 líneas siguientes, salvo si el programa se para antes. La actualización de esta variable del sistema es dinámica. Su utilización no tiene un efecto duradero.

Haga el programa:

```
10 FOR n=1 TO 77
20 PRINT PEEK 23692
30 NEXT n
```

Verá 22 veces el número 1, después scroll?

Pulse una tecla (ni N ni SPACE)
Verá 22 números, 1 después 22 a 2, después scroll?

Pulse una tecla
Verá lo mismo.

Pulse una tecla
Verá que el descuento se para en 13: el programa ha terminado.

Haga PRINT PEEK 23692 y obtendrá 23. El ordenador descuenta de 23 a 1 para visualizar 22 líneas, cuando se responde afirmativamente a scroll?

Añada la línea:

```
5 POKE 23692,200
```

y verá que el ordenador no se detiene para solicitarle scroll? Cuando el programa termina, el último número visualizado es 146.

Haga PRINT PEEK 23692 y obtendrá 23.

23693 ATTR–P

contiene los atributos de color pemanentes.

Los atributos referidos son el color tinta, el color papel, el brillo y el parpadeo (tal como se explica en la función ATTR).

Para comprender lo que sigue, expliquemos lo que es un octeto. Un octeto es una célula de la memoria, que contiene 8 bits numerados de 0 a 7. Cada bit del octeto posee un valor propio.

Cada octeto tiene su propia dirección.
bit 0 valor 1
bit 1 valor 2
bit 2 valor 4
bit 3 valor 8
bit 4 valor 16
bit 5 valor 32
bit 6 valor 64
bit 7 valor 128

Puesto que el ordenador trabaja en binario, usted no se extrañará de que los valores sucesivos sean potencia de 2.

Los bits 0, 1, 2 tienen el color tinta = 1 vez color tinta
Los bits 3, 4, 5 tienen el color papel = 8 veces color papel
El bit 6 tiene el brillo = 64 veces brillo (0 o 1)
El bit 7 tiene el parpadeo = 128 veces el parpadeo (0 o 1)
Si usted desea tinta verde, papel rojo y brillo, o sea 1 * 4 + 8 * 2 + 64 * 1, haga

```
POKE 23693,84
PRINT "VEA"
```

23694 MASK–P

contiene los 4 atributos de la variable del sistema precedente, cuando se declaran transparentes.

Esta variable del sistema oculta, en tal caso, los colores permanentes.

23695 ATTR–T

contiene los atributos de color temporales, puestos por los artículos de color en instrucciones tales como PRINT, PLOT, etc.

23696 MASK–T

contiene los atributos declarados temporalmente como transparentes. Las 2 variables del sistema de arriba, al ser temporales, no son permanentes. (M. de la Palisse sonríe en su tumba.)

23697 P–FLAG

contiene los indicadores para los atributos OVER, INVERSE, tinta contrastada y papel contrastado.

El bit 0 indica OVER 1 temporal
El bit 1 indica OVER 1 permanente
El bit 2 indica INVERSE 1 temporal
El bit 3 indica INVERSE 1 permanente
El bit 4 indica INK 9 temporal
El bit 5 indica INK 9 permanente
El bit 6 indica PAPER 9 temporal
El bit 7 indica PAPER 9 permanente

Para comprender bien las 5 variables del sistema que acabamos de ver y, en consecuencia, para su buena utilización, haga el programa:

```
10 FOR n=23693 TO 23697
20 PRINT PEEK n
30 NEXT n
```

RUN le dará 56, 0, 56, 0, 0

Así pues, papel blanco, tinta negra.

Todavía una pequeña información para captar el mecanismo: el ordenador vuelve a copiar los atributos permanentes en los atributos temporales, después cambia los atributos temporales siguiendo los artículos de color, si los hubiera y, por último, solamente tiene en cuenta las 3 últimas variables del sistema para elaborar su visualización.

Intente:

```
INVERSE 1
```

RUN dará 56, 0, 56, 0, 12 en video invertido
ENTER dará el programa en video invertido.
Pruebe INVERSE 0, después cambie la línea 20:

```
20 PRINT INVERSE 1;PEEK n
```

RUN dará 56, 0, 56, 0, 4 en video invertido
ENTER visualizará el programa en video normal.
Pruebe los otros atributos, ya sean permanentes o temporales, y entienda la visualización.
Usted podrá, a continuación, modificar los atributos con un solo POKE, sabiendo bien lo que hace. No haga POKE en las variables del sistema temporales, ya que sólo se utilizan una vez para un artículo de color situado en las instrucciones que las aceptan: INPUT, PRINT, DRAW, etcétera.

23728/9

estos 2 octetos no son utilizados por el ordenador.

23730/1 RAMTOP

facilita la dirección del último octeto de la zona del sistema Basic. Desde esta dirección y hasta el final de la memoria viva, todos los octetos están protegidos durante una misma sesión. Así, si usted desea introducir códigos máquina, datos para conservar o para transferir un nuevo juego de caracteres, caracteres gráficos, etc., deberá introducir una nueva dirección en RAMTOP (sobre todo con la instrucción CLEAR), después hacer NEW y, por último, empezar su sesión de trabajo.

Exactamente debajo de RAMTOP, existe una zona para la pila de los GO SUB, a continuación otra para la pila del microprocesador. Por esta razón, debemos disminuir RAMTOP y después efectuar NEW, que reorganiza estas 2 pilas.

23732/3 P–RAMT

proporciona la dirección del último octeto de la memoria viva. Es muy poco útil, ya que usted ya sabe si su Spectrum es un 16 k o un 48 k. La dirección del último octeto será 32767 o 65535, respectivamente.



Conozca el ZX-Spectrum
Con el "ZX-Spectrum para todos" en mano, colóquese delante de su máquina y empiece a escribir algunas instrucciones.

Asimilará enseguida las nociones fundamentales de la programación: variables, comprobaciones, circuitos...

Entonces podrá abordar la "magia" del ZX-Spectrum los **gráficos** y los **sonidos**, gracias a los numerosos ejemplos ilustrados y a los programas directamente comentados.

A partir de aquí, le será fácil profundizar sus conocimientos y escribir sus propios programas de gestión, enseñanza, juegos...



EDICIONES ELISA, S. A.

84-7622-008-1